

14

AFIT/GCS/EE/78-6 ✓

1

6

SOFTWARE DESIGN FOR A VISUALLY-COUPLED  
AIRBORNE SYSTEMS SIMULATOR (VCASS) .

9 Master's THESIS,

10

AFIT/GCS/EE/78-6

William H./Reeve

Captain USAF

Jerry L./Stinson

Captain USAF

11

Mar 78

12

248p.

DDC

RECEIVED  
JUN 19 1978

JE

Approved for public release; distribution unlimited.

Φ12 225

act

PII Redacted

SOFTWARE DESIGN FOR A VISUALLY-COUPLED  
AIRBORNE SYSTEMS SIMULATOR (VCASS)

THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science

by

William H. Reeve, B.S.

Captain USAF

Graduate Electrical Engineering

and

Jerry L. Stinson, B.S.

Captain USAF

Graduate Electrical Engineering

March 1978

ACCESSION for		
NTIS	White Section	<input checked="" type="checkbox"/>
DDC	Buff Section	<input type="checkbox"/>
UNANNOUNCED		<input type="checkbox"/>
JUSTIFICATION.....		
BY.....		
DISTRIBUTION/AVAILABILITY CODES		
Dist.	AVAIL. and/or	SPECIAL
A		



## Preface

This thesis describes the application of Structured Analysis and Structured Design methodologies to design the software to simulate a specialized and complex aircraft simulator. The key to the successful development of a software design is a complete and understandable requirements definition. Structured Analysis facilitates the building of a requirements definition that gradually exposes the detail of the system and is well structured. Using this well-structured requirements definition and the Structured Design methodology, a software structure is designed that makes coding, debugging, and modification easier and faster by reducing complexity.

We wish to thank the personnel of the 6570 Aerospace Medical Research Laboratory (AMRL) and the System Research Laboratory (SRL) who have helped to make this thesis a success. In particular we wish to thank Mr. Tom Furness and Mr. Dean Kocian for providing us with the opportunity to work on this research project, Ms. Garnet Borrer for her support and guidance, and Mr. Charles Ballinger for the outstanding graphics contained in this thesis. Thanks are also due to our thesis advisor, Captain J. B. Peterson, for the support he gave us and the time and expertise he shared with us on this endeavor. A very special thanks goes to our wives, Carol and Delois, and our children for their patience and love during this time.

William H. Reeve  
Jerry L. Stinson

## Contents

	<u>Page</u>
Preface . . . . .	ii
List of Figures . . . . .	v
List of Tables . . . . .	viii
Abstract . . . . .	ix
I. Introduction . . . . .	1
Background . . . . .	1
Objectives . . . . .	4
Scope . . . . .	5
Plan of Development . . . . .	6
II. Requirements Definition . . . . .	7
Introduction . . . . .	7
Design Constraints . . . . .	7
Functional Specifications . . . . .	8
Summary . . . . .	15
III. Formal Functional Specifications . . . . .	16
Introduction . . . . .	16
Activity Model . . . . .	17
Data Model . . . . .	76
Summary . . . . .	92
IV. Software Design . . . . .	93
Introduction . . . . .	93
Bubble Chart . . . . .	94
Structure Charts . . . . .	97
Summary . . . . .	176
V. Design Development . . . . .	177
Introduction . . . . .	177
Requirements Definition Phase . . . . .	177
Software Design Phase . . . . .	181
Observations and Recommendations . . . . .	190
Bibliography . . . . .	192



## Contents

	<u>Page</u>
Appendix A: Structured Analysis Diagrams . . . . .	193
Appendix B: Structured Design Charts . . . . .	198
Vitae . . . . .	201

## List of Figures

<u>Figure</u>		<u>Page</u>
1	Visually-Coupled Airborne System Simulator . . . . .	3
2	FSD/Simulated HUD Displays . . . . .	12
3	Vehicle Velocity Hemispherical Display (VVHD) . . . . .	13
4	Simulate VCASS . . . . .	18
5	Simulate VCASS . . . . .	20
6	Process User Commands . . . . .	22
7	Get Valid Command . . . . .	24
8	Configure System . . . . .	26
9	Give Terminal Response . . . . .	28
10	Output Operational Recording . . . . .	30
11	Produce Plant Dynamics . . . . .	32
12	Process Inputs . . . . .	34
13	Compute Vehicle Plant Dynamics . . . . .	36
14	Compute Target Plant Dynamics . . . . .	38
15	Control Semi-Smart Mode . . . . .	40
16	Control Smart Mode . . . . .	42
17	Compute Weapons Plant Dynamics . . . . .	44
18	Compute Attack Performance . . . . .	46
19	Output Operational Recording . . . . .	48
20	Update Aircraft Displays . . . . .	50
21	Process Inputs . . . . .	52
22	Update Symbology Displays . . . . .	54
23	Format Inputs . . . . .	56
24	Makeup Table/Menu . . . . .	58



## List of Figures

<u>Figure</u>		<u>Page</u>
25	Makeup HMD . . . . .	60
26	Makeup HUD Display (HMD) . . . . .	62
27	Makeup HUD . . . . .	64
28	Makeup Panel Displays . . . . .	66
29	Makeup HUD Display (Panel) . . . . .	68
30	Update Imagery Displays . . . . .	70
31	Output Displays . . . . .	72
32	Output Operational Recording . . . . .	74
33	Simulator Data . . . . .	77
34	Simulator Data . . . . .	79
35	Configuration Data . . . . .	82
36	Analog/Digital Inputs . . . . .	84
37	Plant Data . . . . .	86
38	Performance Data . . . . .	88
39	Simulation Displays . . . . .	90
40	Final Bubble Chart . . . . .	95
41	Top Level Structure . . . . .	98
42	PERFORM COMMAND Branch . . . . .	102
43	GET VALID COMMAND Branch . . . . .	106
44	CONFIGURE SIMULATION Branch . . . . .	109
45	PRODUCE PLANT DYNAMICS Branch . . . . .	114
46	PROCESS DISCRETE PLANT INPUTS Branch . . . . .	118
47	COMPUTE VEHICLE DYNAMICS Branch . . . . .	121
48	COMPUTE TARGET DYNAMICS Branch . . . . .	126

## List of Figures

<u>Figure</u>		<u>Page</u>
49	COMPUTE WEAPONS DYNAMICS Branch . . . . .	130
50	COMPUTE ATTACK PERFORMANCE Branch . . . . .	134
51	MAKEUP DISPLAYS Branch . . . . .	138
52	PROCESS DISCRETE DISPLAY INPUTS Branch . . . . .	143
53	MAKEUP TABLE/MENU Branch . . . . .	146
54	MAKEUP HMD Branch . . . . .	150
55	MAKEUP HUD Branch . . . . .	155
56	MAKEUP PANEL DISPLAY Branch . . . . .	158
57	UPDATE COCKPIT INSTRUMENTS Branch . . . . .	162
58	PUT RECORDING Branch . . . . .	165
59	PUT USER MESSAGE Branch . . . . .	168
60	GET PARAMETERS Branch . . . . .	171
61	PUT RECORDING BUFFER Branch . . . . .	174
62	Preliminary Bubble Chart . . . . .	183
63	"First-Cut" Top Level Structure Chart . . . . .	185
64	Alternate CONFIGURE SIMULATION Branch . . . . .	187
65	Box/Interface Arrow Conventions . . . . .	194
66	Mechanism Call Arrow . . . . .	195
67	OR Branch and Join Structure . . . . .	196
68	ICOM Numbering Convention . . . . .	197
69	Information Flow . . . . .	199
70	Information Arrow Types . . . . .	200
71	Decision and Iteration . . . . .	200



## List of Tables

<u>Table</u>	<u>Page</u>
I Parameters for Top Level Structure . . . . .	99
II Parameters for PERFORM COMMAND Branch . . . . .	103
III Parameters for GET VALID COMMAND Branch . . . . .	107
IV Parameters for CONFIGURE SIMULATION Branch . . . . .	110
V Parameters for PRODUCE PLANT DYNAMICS Branch . . . . .	115
VI Parameters for PROCESS DISCRETE PLANT INPUTS Branch . . . . .	119
VII Parameters for COMPUTE VEHICLE DYNAMICS Branch . . . . .	122
VIII Parameters for COMPUTE TARGET DYNAMICS Branch . . . . .	127
IX Parameters for COMPUTE WEAPONS DYNAMICS Branch . . . . .	131
X Parameters for COMPUTE ATTACK PERFORMANCE Branch . . . . .	135
XI Parameters for MAKEUP DISPLAYS Branch . . . . .	139
XII Parameters for PROCESS DISCRETE DISPLAY INPUTS Branch . . . . .	144
XIII Parameters for MAKEUP TABLE/MENU Branch . . . . .	147
XIV Parameters for MAKEUP HMD Branch . . . . .	151
XV Parameters for MAKEUP HUD Branch . . . . .	156
XVI Parameters for MAKEUP PANEL DISPLAY Branch . . . . .	159
XVII Parameters for UPDATE COCKPIT INSTRUMENTS Branch . . . . .	163
XVIII Parameters for PUT RECORDING Branch . . . . .	166
XIX Parameters for PUT USER MESSAGE Branch . . . . .	169
XX Parameters for GET PARAMETERS Branch . . . . .	172

## Abstract

*Abstract* → This thesis contains an analysis of a Visually Coupled Airborne Systems Simulator (VCASS) and the design of the software for this system. The design is developed in three steps. First, an informal requirements definition is written to establish the viewpoint and the purpose on which the analyst bases his design. This requirements definition explains why the simulator is to be created and what it is to do. Second, a top-down strategy called "structured analysis" is applied to obtain a formal requirements definition. The structured analysis is presented in a blueprint-type language consisting of activity and data models. These models represent graphically the functions performed by the simulator and the information upon which those functions act. Third, a design is obtained through a structured design methodology consisting of "transform analysis" and "transaction analysis" techniques. The structure charts drawn during the analysis phase reveal system characteristics which illustrate design quality. The activity model is used to make a successful transition from a top-down analysis to a structured design which can be evaluated. The resulting simulator design, with minor revisions, satisfies the design goals established for the project. The methodologies used are highly recommended for the analysis and design of any software system.

*Abstract* →



SOFTWARE DESIGN FOR A VISUALLY-COUPLED  
AIRBORNE SYSTEMS SIMULATOR (VCASS)

I. Introduction

Background

Because of increasing costs for fuel and aircraft, the Air Force has been forced to cut the number of flight hours of pilot training in the aircraft. As planes become more complex, the pilots require more training to maintain their flying proficiency. The Air Force is, therefore, faced with the problem of obtaining more training for the pilots while still keeping the number of actual aircraft training flights to a minimum.

A present solution is the fixed-based flight simulator. This simulator utilizes electro-optical devices to project video imagery (generated from a sensor scan of a terrain board or from computer-generated imagery) onto a hemispherical dome (or set of large adjacent CRT displays arranged in optical mosaics). Most types of simulators do not incorporate infinity optics or provide collimated visual scenes to the operator. Those which do are large and expensive. The fixed-based simulator, also, does not allow the flexibility of incorporating other display design factors, such as different head-up display formats, variable fields-of-view, representative cockpit visibilities, and optional control and display interfaces.

The visually coupled systems (VCS) is an approach to solving the visual presentation problems of aircraft simulators. VCS includes a combination of a helmet-mounted sight (HMS), eye position sensing system

(EPS), and helmet mounted displays (HMD). It is through the VCS concepts that the idea of a Visually Coupled Airborne Systems Simulator (VCASS) came into being.

VCASS is a flexible, visual scene simulator providing synthesized out-of-the cockpit visual scenes and targets. It is also used to represent an aircraft whose type, threat, and weapons dynamics can be altered with flexibility of control and display configurations. VCASS can be used to simulate either air-to-ground weapons delivery or air-to-air engagement scenarios.

A system block diagram of the functional elements required to accomplish VCASS is shown in Figure 1. The operator utilizes conventional control devices (control stick, throttle, rudder pedals, etc.) as input to a digital computer which provides the manipulation of the vehicle, threat, and weapon states as a function of preprogrammed dynamic characteristics. This input is used to manipulate generated symbology and imagery in terms of aircraft orientation, scale, target location, etc. A visual scene (generated by the graphics or sensor imagery generators) is selected by the operators line-of-sight orientation, as measured by the helmet-mounted sight sensors. The helmet display electronics receives the selected portion of the symbology and sensor information and displays the video imagery to the operator through the helmet display optics, in the proper orientation within three-dimensional space. The operator engages electronic targets (either air-to-air or air-to-ground) and launches electronic weapons. The operator performance is assessed and recorded for future reference.



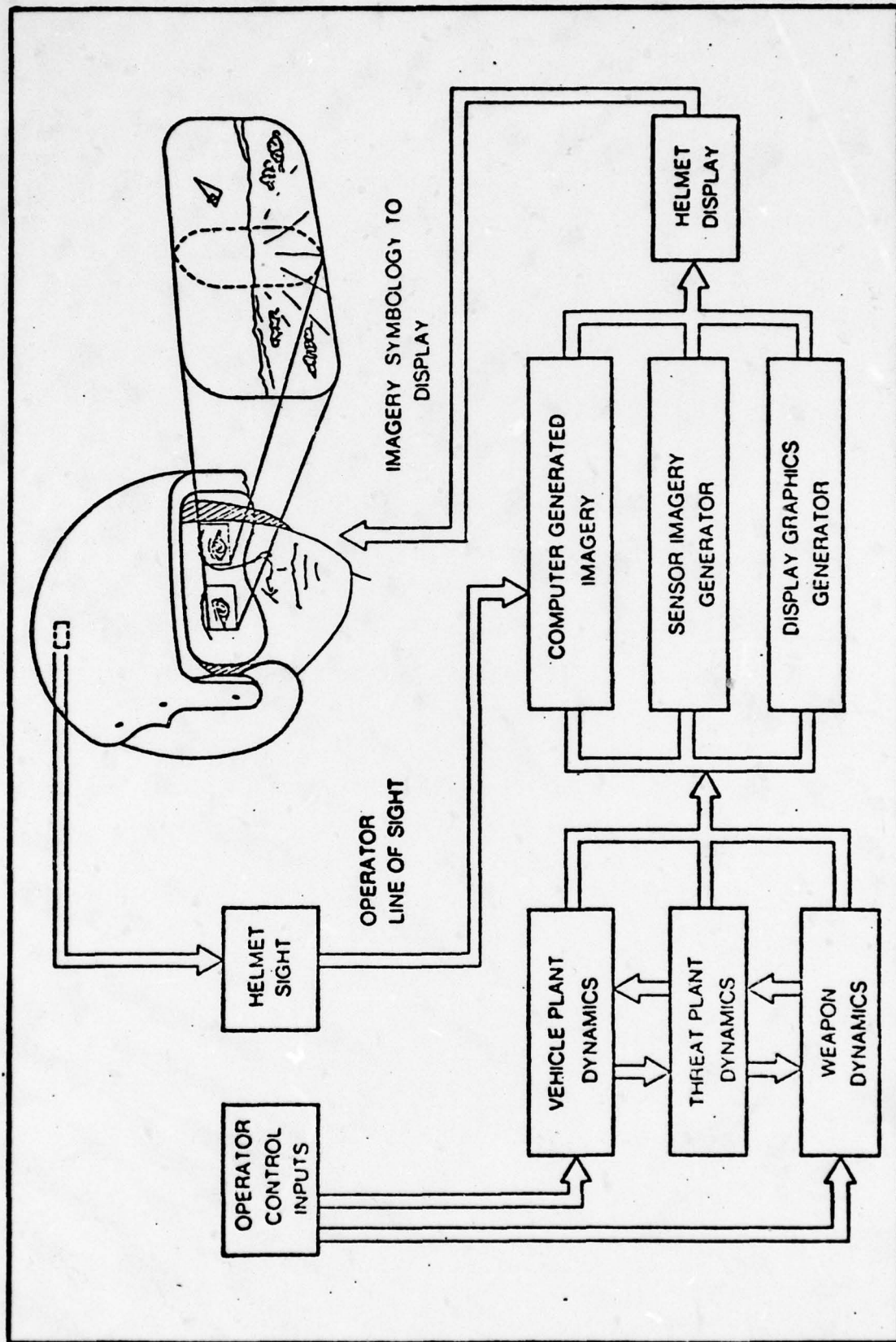


Figure 1. Visually-Coupled Airborne Systems Simulator

## Objectives

This thesis is concerned with designing the software to accomplish the VCASS project objectives. The software design involves defining the necessary subsystems, programs or modules, and their interconnections. In order to develop the software design, it is first necessary to define the functional specifications of the system. When the functional specifications are not well defined, a premature and poorly designed system, leading to skyrocketing costs, missed schedules, waste, duplication, and disgruntled users, often results (Ref 5:2). The primary objective of this thesis is to avoid such problems. To meet this objective, a good development technique is required. Structured analysis is such a technique.

Structured analysis is a comprehensive methodology for performing functional analysis and design. For this project, only the functional analysis phase of structured analysis is used. During this phase, the emphasis is on analyzing and documenting "what" the system is supposed to do. Two sets of diagrams result: one describes the system in terms of activities and the other describes the system in terms of data. These diagrams are created by decomposing the system into smaller and smaller pieces. The two sets of diagrams are cross-checked and sequenced to provide a model of the system functions. This model provides the base for the design (Ref 6: 1-1, 1-2).

After the functional specifications are defined, the software structure is designed through the application of structured design techniques. Structured design is a set of general program design considerations and techniques for making coding, debugging and modification easier, faster, and less expensive (Ref 8:114). It simplifies the



software by dividing it into modules in such a way that modules can be implemented and modified with minimal consideration or effect on the other parts of the system. A graphical tool, called structured charts, is used to present the software structured design. The interface between the modules is given, along with a functional description of each module.

The sponsor of this thesis is the 6570 Aerospace Medical Research Laboratory (AMRL), located at Wright-Patterson Air Force Base, Ohio. Their mission has been to optimize the visual interface of crew members to advanced weapon systems. This mission has been primarily pursued in two areas: (1) the establishment of control and display engineering criteria, and (2) the prototyping of advanced concepts of control and display interfaces. The development of an operational VCASS system will be an important step in fulfilling this mission.

### Scope

The purpose of this thesis is to develop a software design that meets the requirements of the VCASS concept. In addition to the normal VCASS functions, this software design will allow for a systematic investigation of display symbology and line graphics representation of terrain and target features.

This thesis is limited to the software design of the proposed system. The hardware or any associated controls are defined only in terms of the input/output interfaces between the helmet sight/display components, simulator cockpit controls, graphics generators, and any interactive keyboard consoles. The resultant design gives a modular structure identifying all inputs and outputs. These modules when fully

coded, should produce a system that will be modifiable, maintainable, and effective in simulating the VCASS.

#### Plan of Development

The requirements definition with emphasis on the functional specifications is discussed in Chapter II. Chapter III presents the formal functional specifications for the system. These specifications are in the form of activity and data models. They are the result of the functional analysis phase of Softech's Structured Analysis and Design Technique (SADT). Chapter IV presents the software design. The design is illustrated with structured charts and data flow graphs (bubble charts) along with the functional descriptions of the modules. A discussion of the techniques used in the design of the system along with conclusions and recommendations are contained in Chapter V.

## II. Requirements Definition

### Introduction

The normal life-cycle for a computer software development project consists of the following seven phases (Ref 2: 5-8): conceptual, requirements definition, design, coding, checkout, testing, and operational. The phase most often underutilized, or even neglected, is the requirement definition. A poor requirements definition usually results in rising costs, missed schedules, waste, duplication, and dissatisfied users. (Ref 5:2). Neglect in this phase often results in an incomplete documentation package. This lack of documentation leads to more wastage when a system has to be "redeveloped" for a modification change.

In this thesis the requirements definition deals with three interrelated subjects: context analysis, design constraints, and functional specifications. The context analysis tells why the VCASS system is being created; this is discussed in the background section of Chapter I. The design constraints tell how the VCASS system is to be constructed; these are discussed in the next section of this chapter. The functional specifications describe what the system is to do and are discussed informally in a later section of this chapter and formally in Chapter III.

### Design Constraints

This section is a summary of the conditions specifying how the VCASS software is to be constructed. No particular hardware, such as input or output devices, will be specified since these details should be specified in a later stage of the design (Ref 5:4).



The four accepted goals of the discipline of software engineering are modifiability, efficiency, reliability, and understandability. (Ref 1:92). These goals were the constraints used in selecting the analysis and design techniques for the design of the VCASS software. The techniques selected are SofTech's structured analysis for the formal functional specifications and Yourdon and Constantine's design method for the software design.

### Functional Specifications

The system to be designed is to simulate VCASS. It should accept inputs from an aircraft representative vehicle, whose type, threat, and weapons dynamics can be altered; it should provide synthesized out-of-the cockpit instrumentation, visual scenes, and targets.

The software portion will have two basic modes of operation, Exerciser and Operational. The Exerciser mode allows the user to interact with the system to build and modify the formats for the aircraft symbology displays. During this mode of operation the system will output messages to the user in response to the options and parameters entered. These symbology displays are then used under the Operational mode to evaluate the pilots perceptibility. The software design for the Exerciser mode of operation is not contained in this thesis but is being designed as a separate subsystem called the Symbology Exerciser of the VCASS simulator. The design of the Symbology Exerciser can be found in Ref 9. This thesis defines the interface between the Symbology Exerciser and the VCASS simulator.

In the Operational mode, the system functions as an aircraft simulator. Flight control and target inputs will be processed, according to pre-set parameters, to update aircraft instruments, symbology displays, and background imagery. There are four main functions that will be

performed in this mode. They are (1) process configuration parameters, (2) produce the plant dynamics, (3) update the aircraft displays, and (4) perform operational recording. The first function is an initialization step only, while the next three functions are continuously repeated in a "real-time" environment. A brief description of each of these functions follows.

Process Configuration Parameters. When the system is put into the Operational mode a message is output to the user requesting configuration parameters. The user then interacts with the system by supplying the required parameters for system options, vehicles, weapons, cockpit, target, environment, and display formats. The user can choose to enter all parameters or only a partial set of parameters. System default parameters are used for all parameters undeclared by the user.

A frequent user of the system will not have to be prompted by the system for the parameters, the system will allow the user to input a command and the parameters at the same time. The system will also be capable of obtaining the parameters from a specified storage device.

Configuration parameters are such things as aircraft type and characteristics, weapons available and quantities, weather conditions, and target characteristics. These parameters are used to update and produce the plant dynamics and simulation displays (aircraft instruments, symbology displays, background imagery, etc.).

Produce Plant Dynamics. After the configuration parameters have been processed, the system enters a "real-time" state and concurrently performs the functions 'Produce Plant Dynamics' and 'Update Aircraft Displays'. The information generated by 'Produce Plant Dynamics' is used by 'Update

Aircraft Displays' to make-up and update the displays. During this "real-time" state, the system processes both discrete and non-discrete inputs. Some of the discrete inputs are vehicle initialization, weapons mode selection, target mode selection, and visually-coupled sight (VCS) control inputs. The non-discrete inputs must be periodically queried to obtain the latest information. Some of these inputs are vehicle control, weapons mode, target mode, and head-mounted-sight (HMS) attitude and position inputs.

The function 'Produce Plant Dynamics' uses both discrete and non-discrete inputs to manipulate the vehicle, weapons, and target states. The updated state variables are then used to compute and to evaluate the attack performance. A description of the plant dynamics and the attack performance follows.

1. Vehicle Dynamics. The vehicle position, angles and velocity are initially entered. Vehicle flight control inputs, such as stick, rudder, and throttle, are then used to compute and update the appropriate vehicle state variables.

2. Weapons Dynamics. The aircraft weapons are activated through one of three modes specified by the operator. The modes are gun, air-air missile, and air-ground. When any of these modes are activated, the aircraft and its associated weapons are setup for release. This setup includes checking the weapons, activating the sights, and determining the lead angles.

3. Target Dynamics. There are three types of targets: canned, semi-smart, and smart. The canned target flight path is based upon a pre-programmed table-lookup. The flight path of the semi-smart target is obtained from a pre-programmed table based upon the attacking vehicles



maneuvers. The smart target is maneuvered by a computer algorithm which is based upon the state of the vehicle and the state of the target.

4. Attack Performance. The two types of scenario to be evaluated are air-air and air-ground. The scenario is determined by the weapons mode. The weapon trajectory and probability of a hit, which are used to evaluate the air-air scenario, are computed using the position, angles, and rate of the aircraft and target. The weapons drag coefficients are also included in this computation.

To evaluate the air-ground scenario the accuracy of the simulated air-to-ground weapons delivery is determined. The known characteristics of the weapons motion and the weapons release point, are used to compute the impact point. This impact point is compared with the known target location to compute a miss distance and angle.

Update Aircraft Displays. The plant dynamics and simulation specific inputs are used to update the cockpit instruments and aircraft displays. The cockpit instruments are similar to the instruments found in an aircraft cockpit and are updated from the respective vehicle state variables. These instruments include the fuel gauge, magnetic compass, airspeed indicator, altimeter and power indicator.

The aircraft displays consist of three different types of hardware display devices, each requiring its independent software package. They are (1) the helmet mounted display (HMD), (2) the panel display, and (3) the heads-up display (HUD). The panel and the HMD, display a field-of-view portion of the outside world display, presenting the proper orientation within a three-dimensional space. The HMD and the panel display are different in that the HMD is oriented with respect to the

operator line-of-sight, while the panel display is oriented with respect to the aircraft heading.

The displays that can be output on the HMD and the panel display are the flight simulated display (FSD), the simulated HUD display, the vehicle velocity hemispherical display (VVHD), the terrain portrayal display, the background-environment display, the target, and the weapons envelope. In addition to the above displays, the HMD also displays the tables and menus. A description of these displays follows.

Since the HUD display can be simulated on the HMD and panel display, the description of the HUD is the same as the description for the simulated HUD in the FSD/simulated HUD description below.

1. FSD/Simulated HUD. Figure 2 illustrates a typical format of the FSD or the simulated HUD display. Some of the information shown in the FSD and in the simulated HUD display are altitude, heading, airspeed,

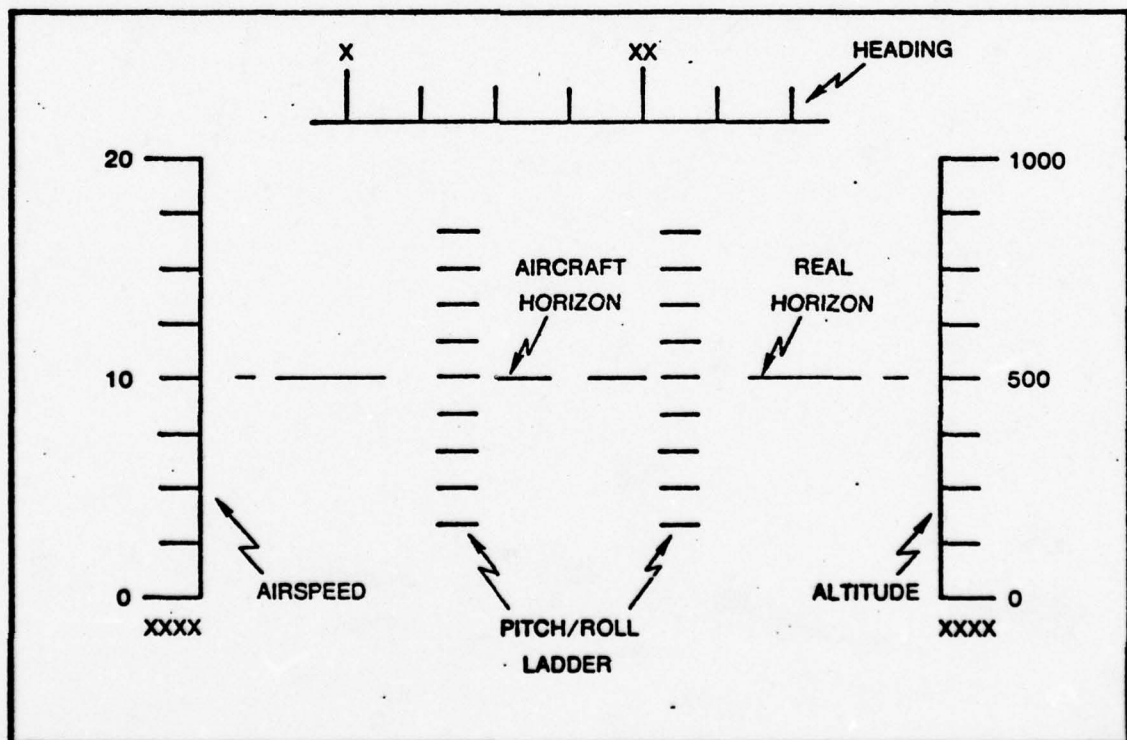


Figure 2. FSD/Simulated HUD Displays

pitch, roll, aircraft horizon, and the real horizon. The FSD and simulated HUD do not always display the same information or have the same format. Both give a graphic representation of the aircraft state, but the HUD changes according to the flight mode while the FSD does not. The flight modes are takeoff, enroute, landing and attack.

2. Vehicle Velocity Hemispherical Display (VVHD). Figure 3 illustrates a typical format of the VVHD display. The simulated horizon lines depict the horizon of the real earth. The converging dashed lines move toward the operator as a function of the vehicle velocity. The amount of convergence or divergence of these lines is a function of the vehicle altitude. This display orients the operator with respect to the vehicle altitude, ground speed, and heading.

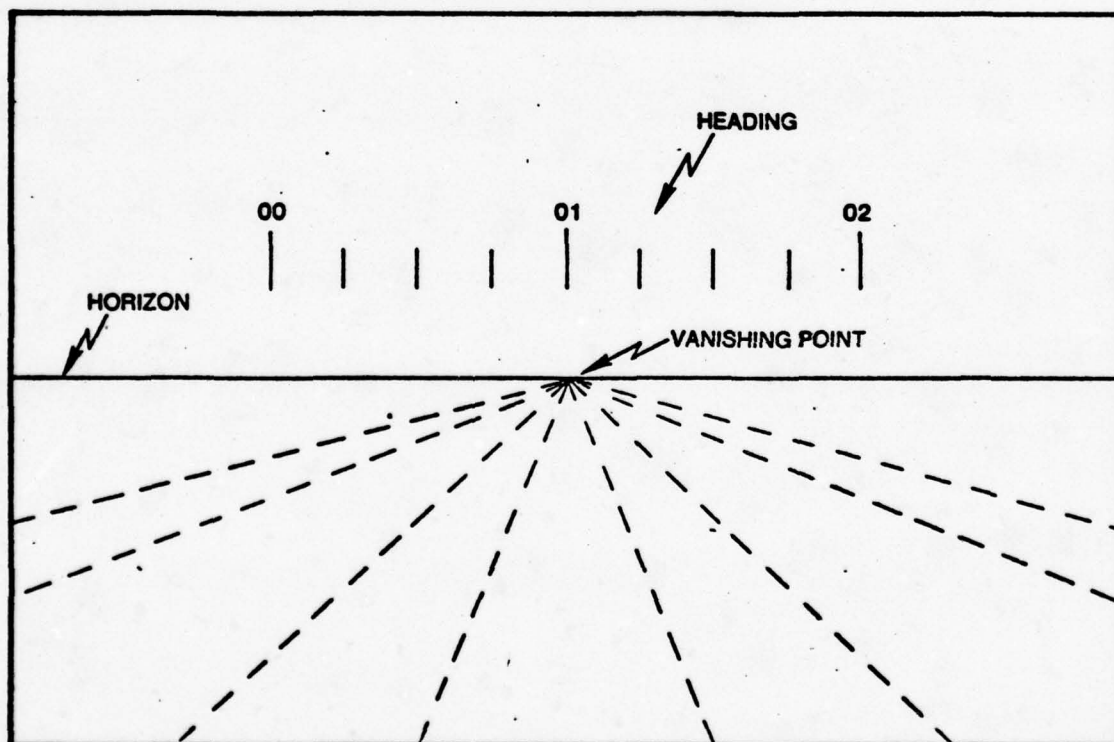


Figure 3. Vehicle Velocity Hemispherical Display (VVHD)



3. Terrain Portrayal. The terrain portrayal is a graphical representation of the earth. This representation is presented as a set of different grid patterns. The particular pattern displayed is a function of the vehicle altitude. This display acts as a reminder to the operator of the vehicle attitude and altitude changes.

4. Background and Environment. The background and environment is a graphical simulation of the world outside the cockpit. It contains graphical features such as mountains, rivers, runways and buildings.

5. Target. The target display is a graphical representation of an aircraft. The target motion is derived from a series of table look-ups, an algorithm, or a combination of both.

6. Weapon Envelopes. The weapons envelope is a graphical representation of the effective radius of the weapon. This display is used by the pilot to tell when a target is within effective range of the selected weapon.

7. Table and Menu. The table and menu displays give the operator the opportunity to select different HMD displays. It also allows the operator to enter, delete, or change system data while in the Operational mode. For example, the FLY-TO-POINTS menu display would be used to enter the coordinates of flight destination points.

Operational Recording. A history of all inputs, outputs, and selected variables is maintained on a recording device. The recording contains such information as hardware errors, operator errors, terminal responses, computed plant dynamics, attack performance, simulation specific inputs, and plant specific inputs. It will be used to evaluate the pilot and access his reactions to various stimuli.

### Summary

Chapter I presented the context analysis of the problem which forms a perspective from which to view the overall problem. This chapter presented the constraints for the problem solution and the informal functional specifications. The design constraints are imposed by the desired design goals. Analysis and design techniques are selected so the design meets these goals. The informal functional specifications in this chapter form the basis for the construction of SADT activity and data models in the next chapter.

### III. Formal Functional Specifications

#### Introduction

The formal functional specifications of the system are developed using the functional analysis phase of Softech Structured Analysis and Design Technique (SADT). The emphasis is on analyzing and documenting "what" the system is supposed to do. Two sets of diagrams result: one type describes the system in terms of activities and the other type describes the system in terms of data. These diagrams are created by decomposing the system into smaller and smaller sections. The two sets of diagrams are cross-checked for consistency to provide a model of the system functions, which make up the formal functional specifications. These diagrams represent the conceptual ideas conveyed by the function specifications, which were discussed in Chapter II.

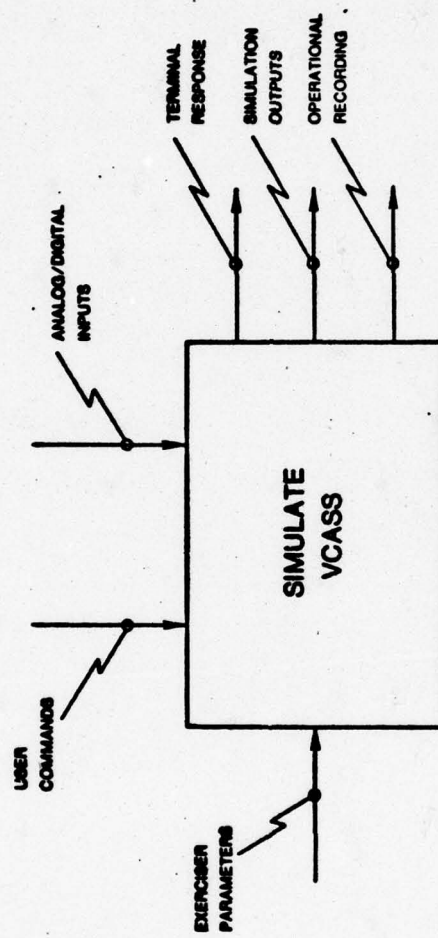
The activity and data models are presented in this chapter and are organized as a sequence of diagrams with the supporting text. The activity model is presented first followed by the data model. Before reading the activity or data diagrams, scan the corresponding "node index", which serves as a table of contents. This index gives an overview of the decomposition structure.

A brief discussion of how to read structured analysis diagrams can be found in Appendix A.



## Activity Model

<u>Node</u>	<u>Title</u>	<u>Page</u>
A-0	Simulate VCASS	18
A0	Simulate VCASS	20
A1	Process User Commands	22
A11	Get Valid Command	24
A13	Configure System	26
A14	Give Terminal Response	28
A15	Output Operational Recording	30
A2	Produce Plant Dynamics	32
A21	Process Inputs	34
A22	Compute Vehicle Plant Dynamics	36
A23	Compute Target Plant Dynamics	38
A233	Control Semi-Smart Mode	40
A234	Control Smart Mode	42
A24	Compute Weapons Plant Dynamics	44
A25	Compute Attack Performance	46
A26	Output Operational Recording	48
A3	Update Aircraft Displays	50
A31	Process Inputs	52
A32	Update Symbology Displays	54
A321	Format Inputs	56
A322	Makeup Table/Menu	58
A324	Makeup HMD	60
A3243	Makeup HUD Display (HMD)	62
A325	Makeup HUD	64
A326	Makeup Panel Displays	66
A3263	Makeup HUD Display (panel)	68
A33	Update Imagery Displays	70
A35	Output Displays	72
A36	Output Operational Recording	74



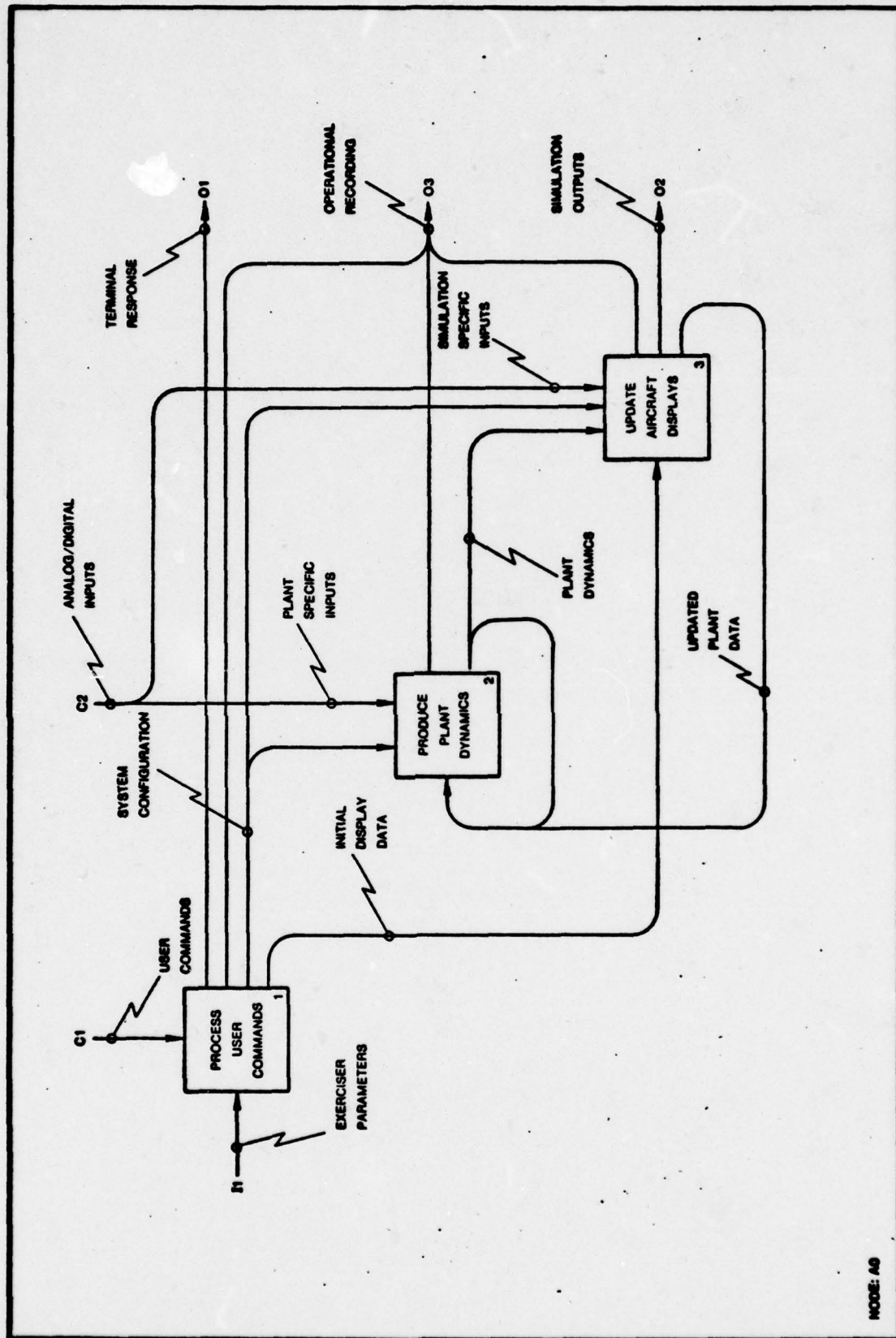
MODE: A-9

Figure 4. Simulate VCASS

**Purpose and Viewpoint:** This activity model provides the formal functional specifications for the VCASS from the viewpoint of a system software designer. The system should accept inputs from an aircraft representative vehicle, whose type, threat, and weapons dynamics can be altered; it should provide synthesized out-of-the cockpit instrumentation, visual scenes, and targets.

**A-0 Text:** The user commands (C1) control the mode of operation of the simulator. There are two basic modes of operation: the Exerciser mode and the Operational mode. During the Exerciser mode, the user is prompted by terminal response (O1) to input set-up parameters (I1). The Operational mode is constrained by the analog and digital inputs (C2), and produces simulation outputs (O2) and operational recording (O3).





MODE: AG

Figure 5. Simulate VCASS

A0 Text: A user command (1C1) is received by 'Process User Commands' (1). The command is either an Exerciser command or an Operational command. If it is an Exerciser command the user is prompted by terminal responses (101) to enter Exerciser parameters (111). The Exerciser parameters are used to set up the initial display data (104).

If the user command (1C1) is an Operational command, parameters are obtained from the command to produce the system configuration (103). The system configuration (2C1) and the plant specific inputs (2C2) are used by (2) to produce the plant dynamics (202).

The plant dynamics (3C1), the system configuration (3C2), and the simulation specific inputs (3C3) are used by (3) to produce the simulation outputs (302) and to update the plant data (303).

A history of all inputs and all actions is recorded (102, 201, 301).

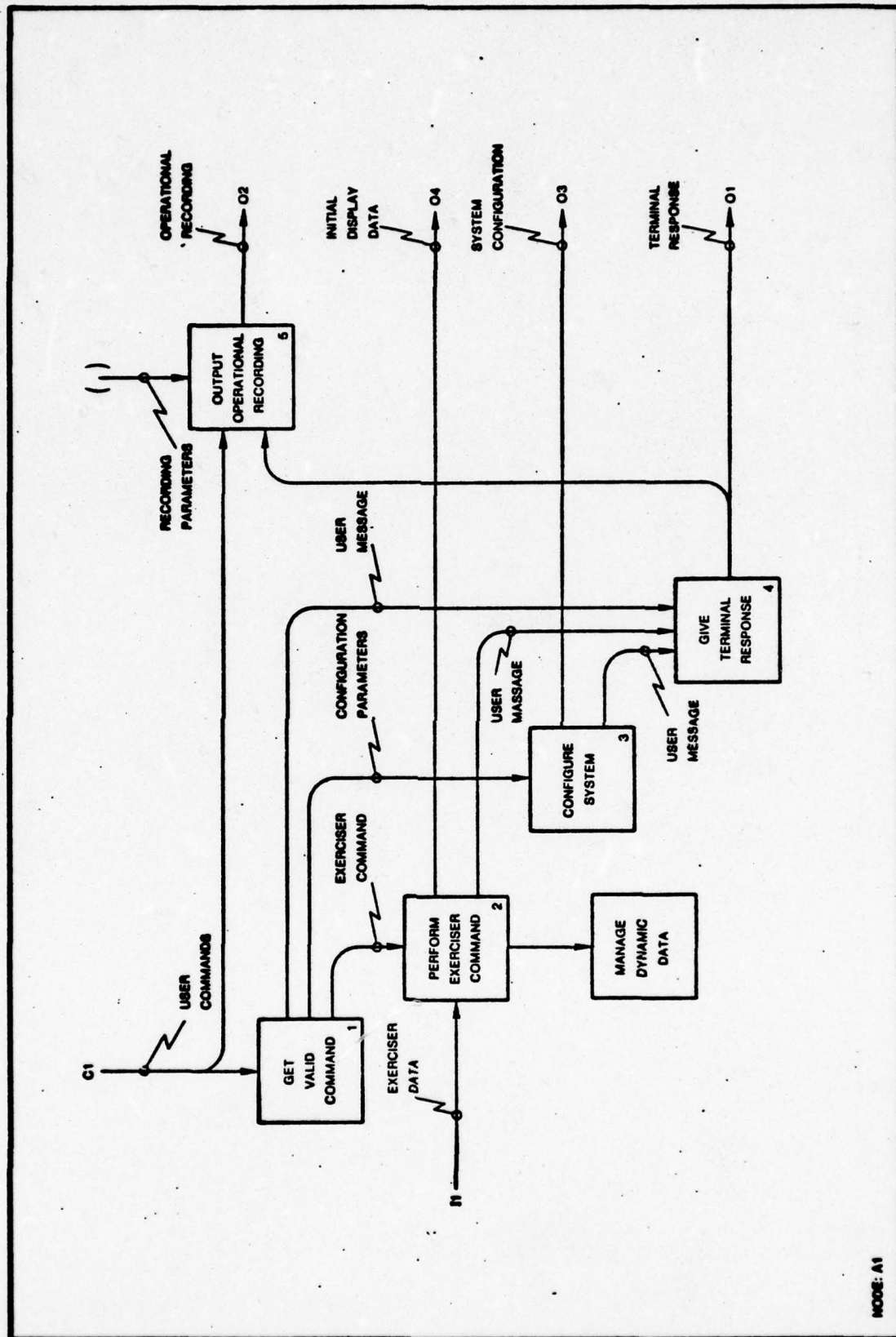


Figure 6. Process User Commands

MODE: A1



AI Text: User commands (1C1) are checked by (1) to determine validity. If they are invalid, an error message (101) is generated. Otherwise, 'Get Valid Command' (1) determines if the command is an Operational Command or an Exerciser Command (103). If the command is an Operational Command 'Get Valid Command' (1) determines if the command parameters have been input. If they have not, it generates a request (101) to the user to input the appropriate parameters. When the parameters (102) are obtained, they are passed to (3).

'Perform Exerciser Command' (2) is a sub-system ('Manage Dynamic Data') which allows for building and modifying formats for aircraft symbology displays. It is initiated through an Exerciser Command (2C1), and allows the user to interact with the system to build the initial display data (201).

The configuration parameters (3C1) are used by (3) to setup the vehicle, weapons, cockpit, target, and environment configurations (301).

The user messages (4C1, 4C2, 4C3), for supplying information to or requesting information from the user, are formatted by (4) for output (401).

All user commands (511) and terminal responses (512) are recorded by (5) as part of the system history (501).

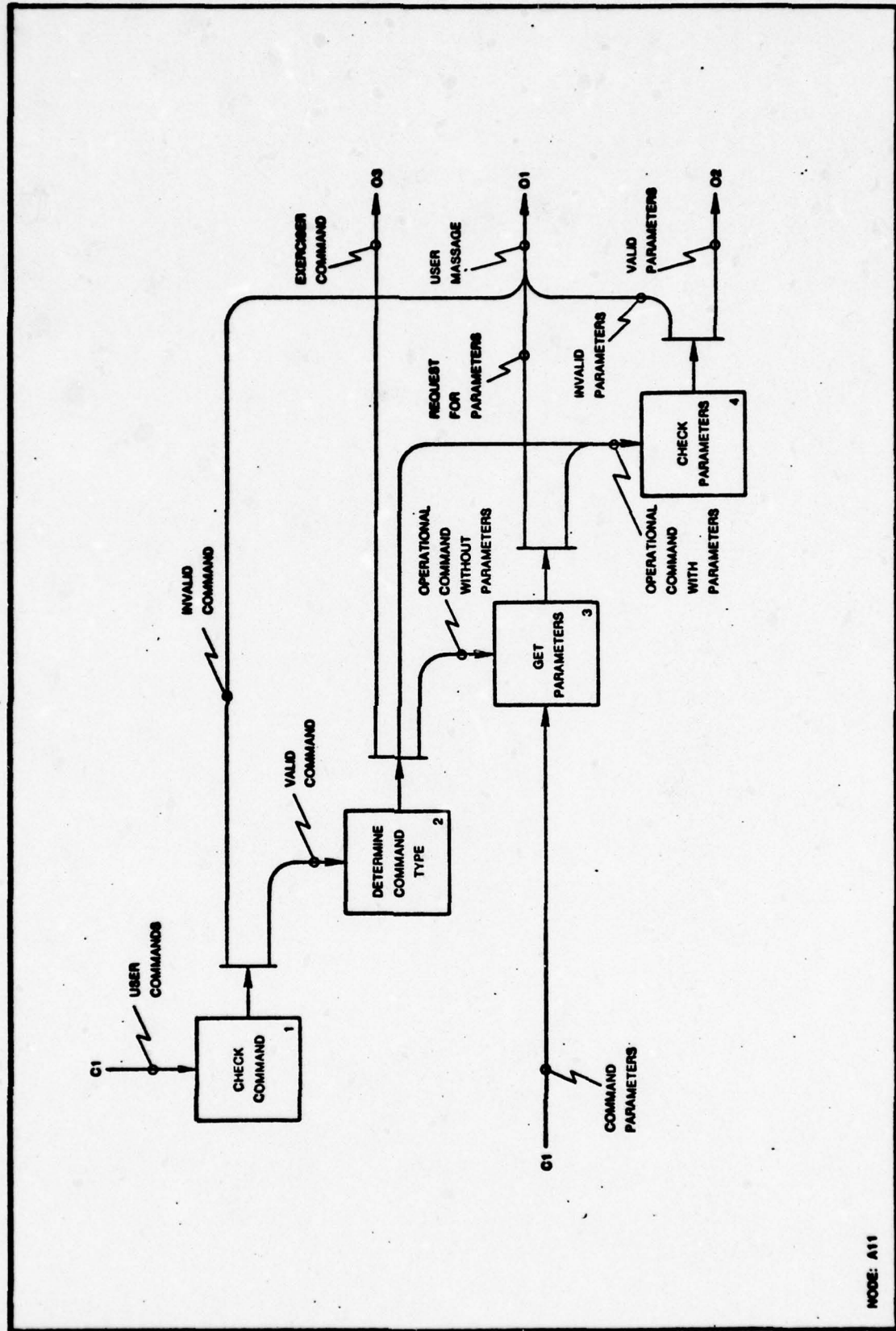


Figure 7. Get Valid Command

MODE: A11

All Text: 'Check Command' (1) checks user commands (101) to see if they are valid. For commands that are invalid, an error message (101) is output. For a valid command (201), the command type is determined by (2); the type is either Exerciser (201) or Operational (202, 203).

For an Operational command which does not have the necessary command parameters (301), 'Get Parameters' (3) outputs a request (301) for the parameters. The parameters are input (311), which results in an Operational command with parameters to be output (302) to 'Check Parameters' (4).

'Check Parameters' (4) receives the Operational command with parameters (401) and checks the validity of the parameters. If an error is found, an error message (401) is generated; otherwise, the parameters (402) are passed and used to configure the system.



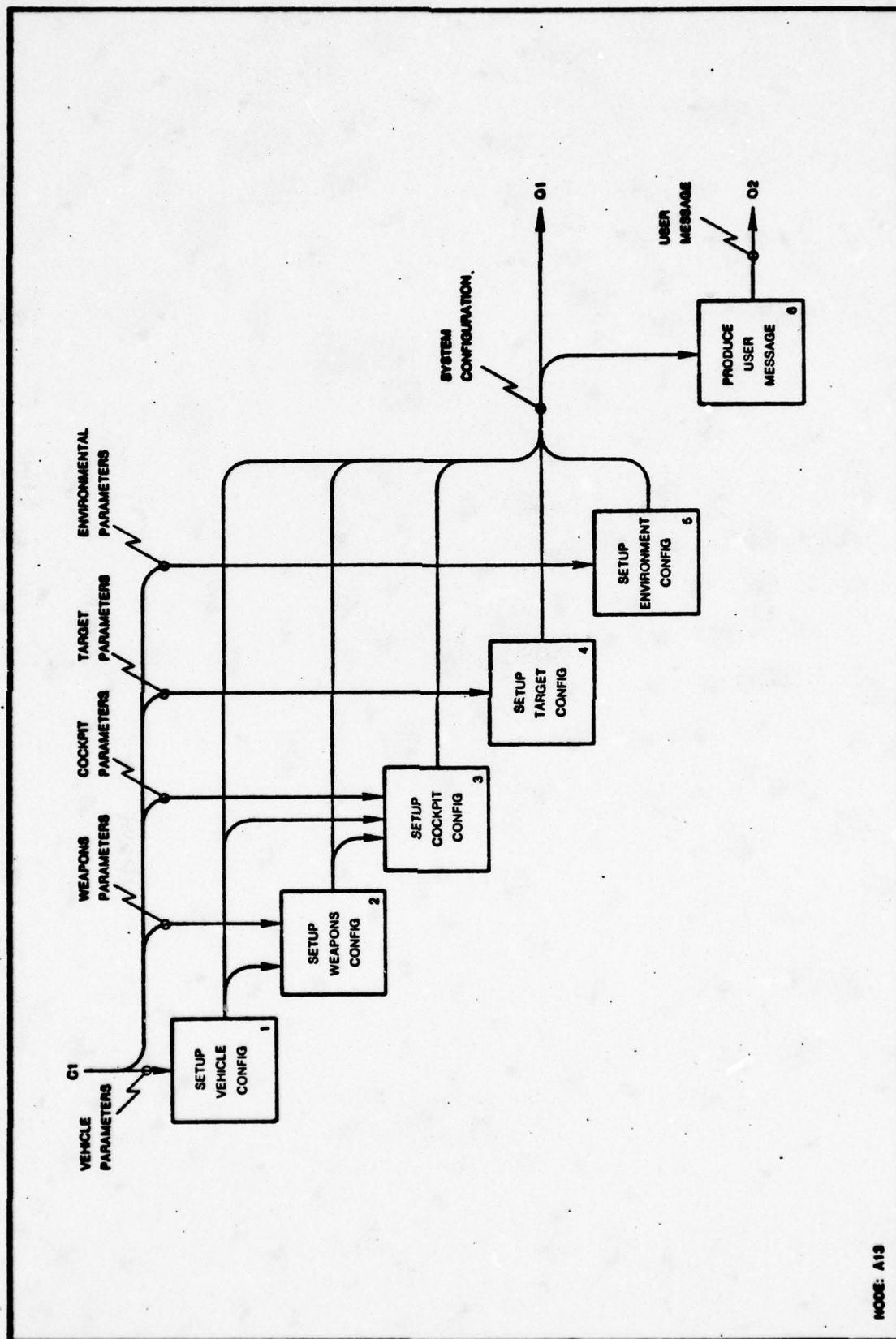


Figure 8. Configure System

MODE: A13

A13 Text: The configuration parameters (1C1, 2C2, 3C3, 4C1, 5C1) are passed to the appropriate activity which sets up the respective part of the system configuration (1C1, 2C1, 3C1, 4C1, 5C1).

Configuration parameters are such things as aircraft type and characteristics (1C1), weapons available and quantities (2C2), target characteristics (4C1), and weather conditions (5C1). These parameters are used to update and produce the plant dynamics and simulation displays.

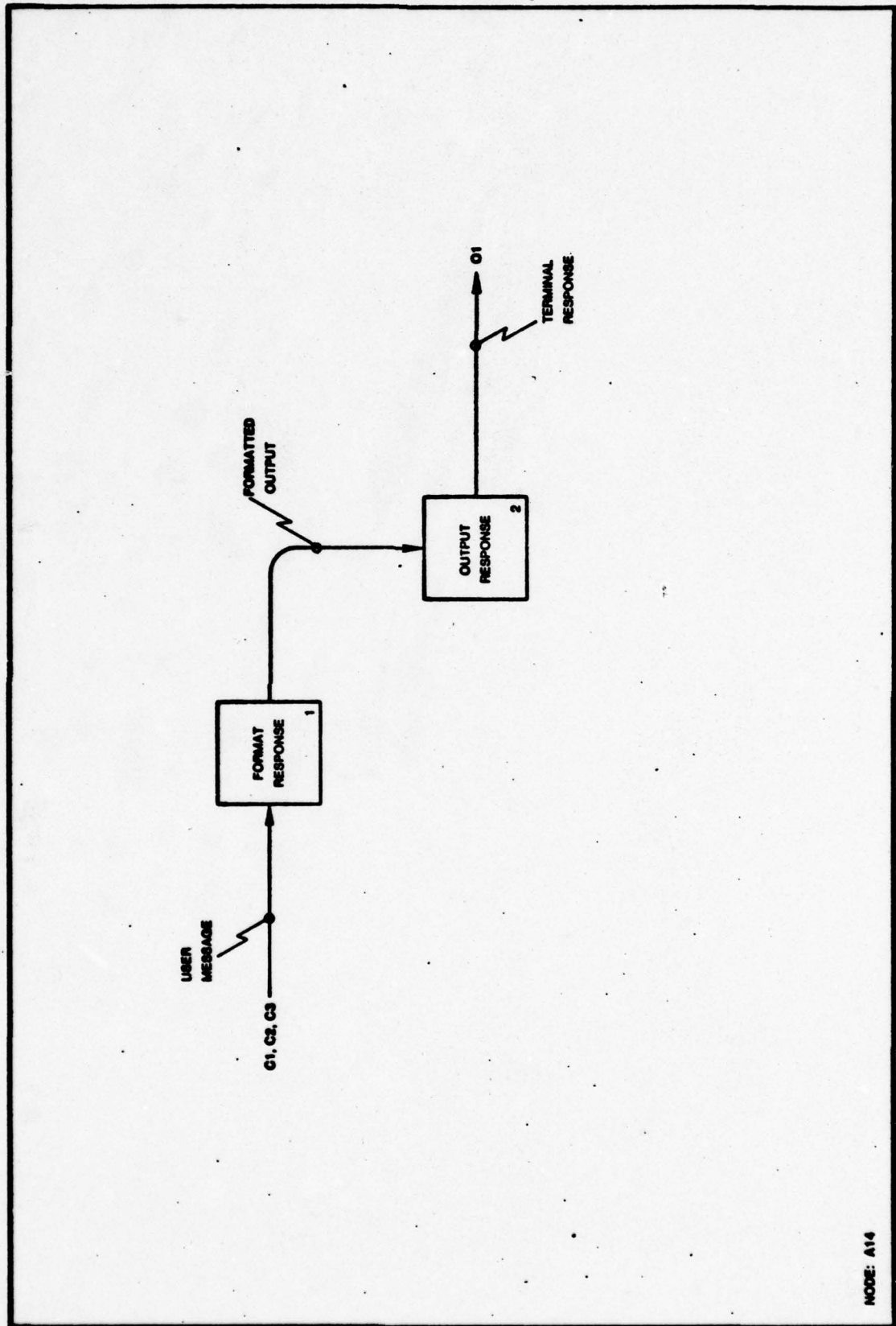


Figure 9. Give Terminal Response



AL4 Text: 'Format Response' (1) formats user messages (111) for output by (2) to the user terminal.

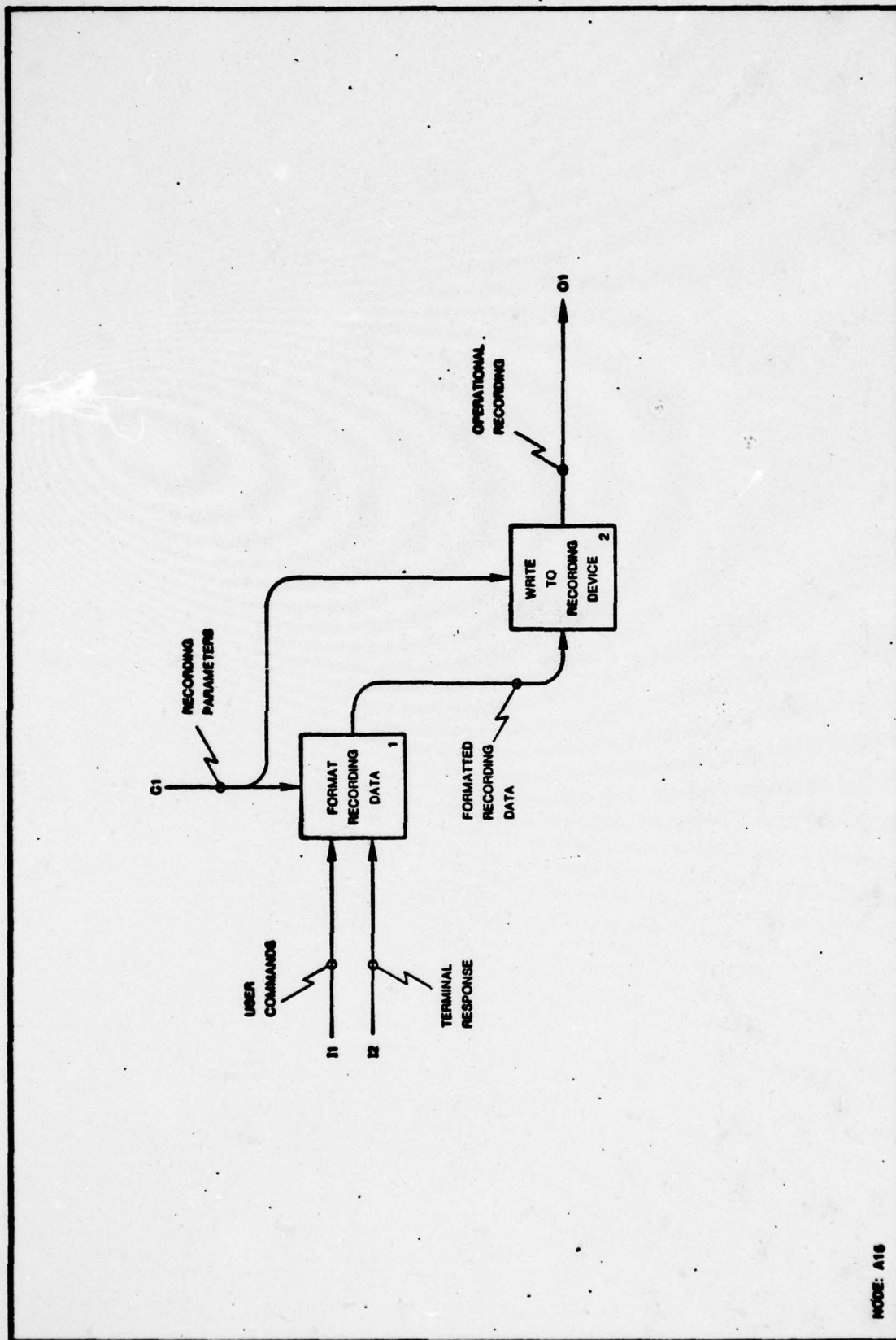
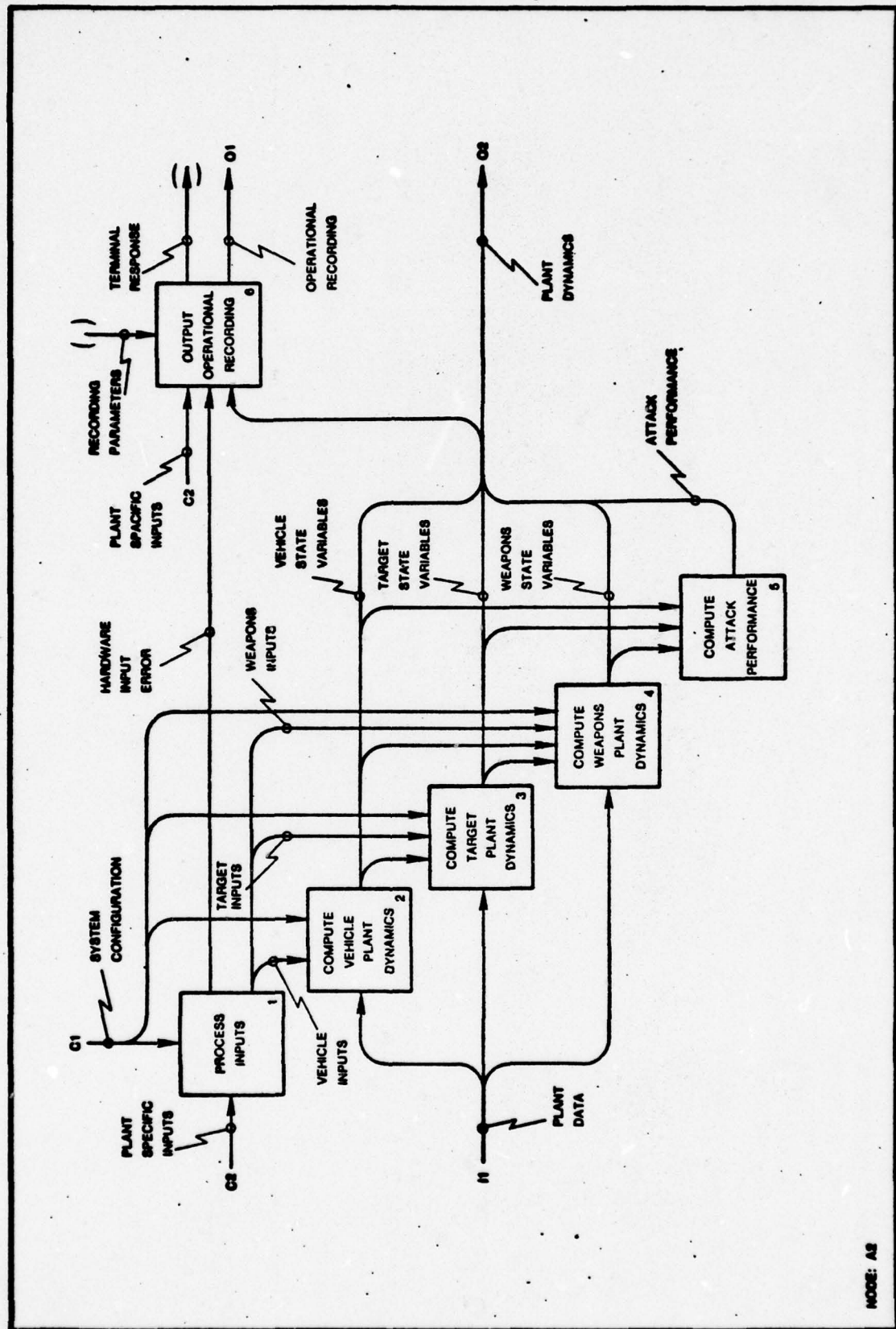


Figure 10. Output Operational Recording

MODE: A16

A15 Text: The user commands (111) and the terminal responses (112) are formatted by (1) into the proper recording format, according to the specifications of the recording parameters (1C1). The formatted recording data (211) are written to a recording device by (2), producing an operational recording (201).





MODE: AS

Figure 11. Produce Plant Dynamics

**A2 Text:** The plant specific inputs (111) (such as throttle, stick and rudder inputs) are received by 'Process Inputs' (1). If there are no hardware input errors the inputs are formatted and used to control the activities (2,3,4). The hardware input errors (101) are recorded as part of the system history (602), and an error message (601) is output to the user by (6).

'Compute Vehicle Plant Dynamics' initializes the vehicle position, rate, and angle; and then receives the vehicle inputs (2C1) to update the vehicle state variables (201).

The target mode is determined by (3) from the target inputs (202). The vehicle state variables (3C1) and the target configuration (3C3) are then used to control the particular target mode by modifying and updating the target state variables (301).

'Compute Weapons Plant Dynamics' (4) determines the weapons mode from the weapons inputs (4C3) and the weapons configuration (4C4). The vehicle state variables (4C2) and the target state variables (4C1) are then used to control the particular weapons mode by modifying and updating the weapons state variables (401). The state variables (5C1, 5C2, 5C3) are used by (5) to compute the attack performance (501). The attack performance (501) contains data about pilot performance in an attack, such as number of hits, adjusted impact point, or miss distance and angle.

The plant specific inputs (611), plant state variables (613), and performance measure (613) are recorded by (6) as part of the system history (602).

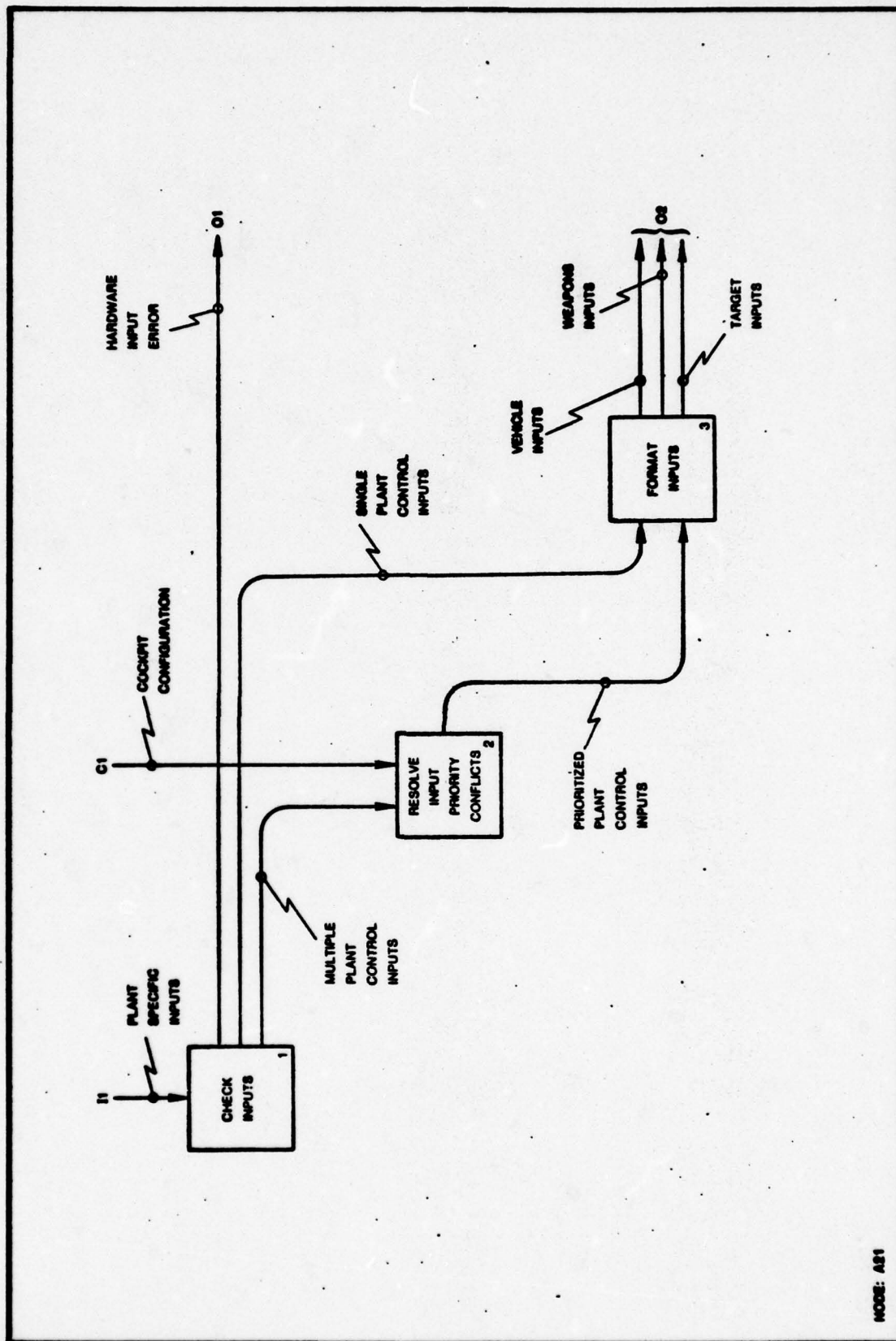


Figure 12. Process Inputs



A21 Text: 'Check Inputs' (1) receives plant specific inputs (101) and checks for validity. If a hardware error is found, a corresponding hardware input error message (101) is generated. Otherwise single plant control inputs (102) are formatted by (3), and multiple plant control inputs (103) are passed to (2). With a dual cockpit configuration, it is possible to receive the same type inputs from both cockpits. These multiple plant control input conflicts (201) are resolved by (2), and the prioritized plant control inputs (201) are formatted by (3).

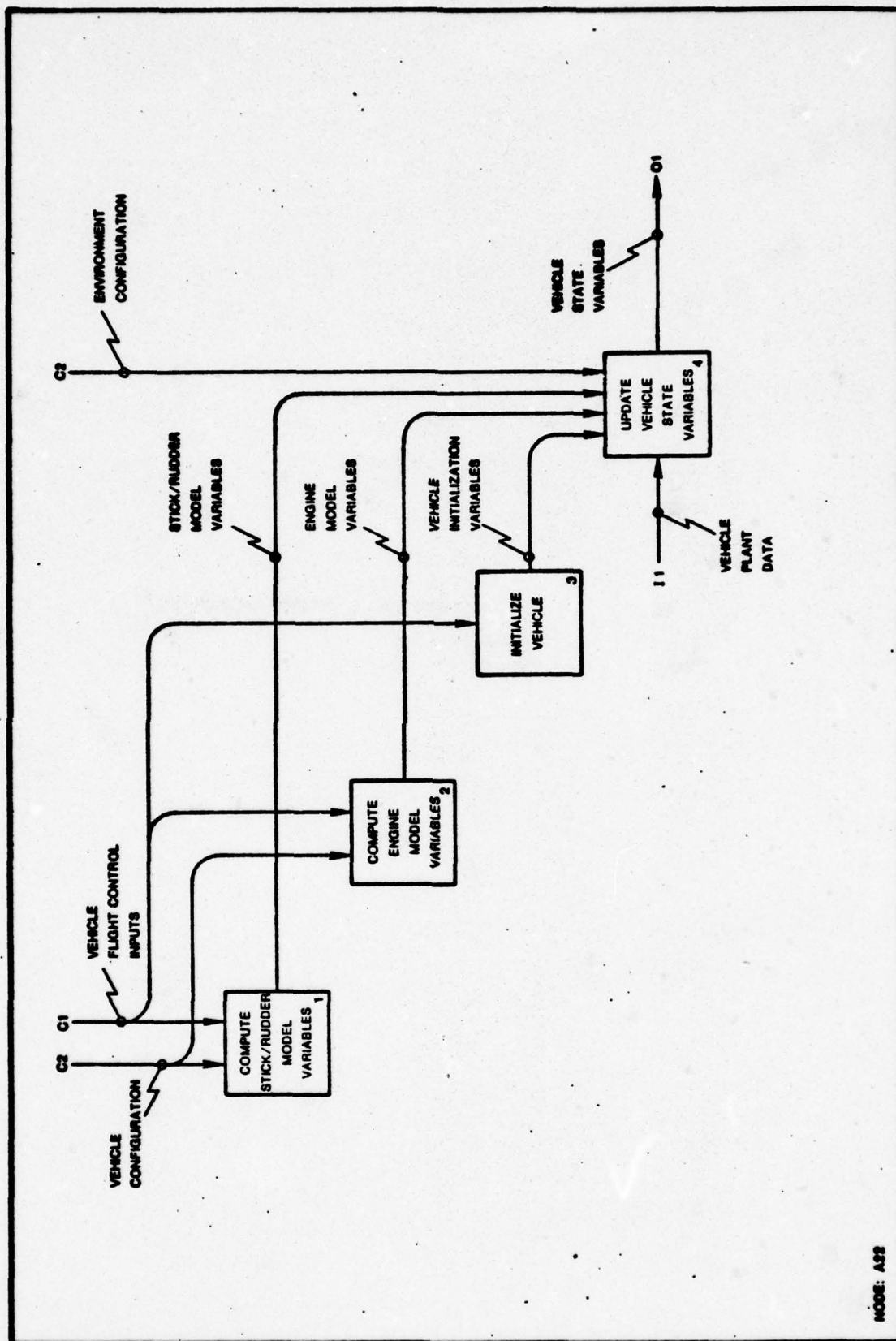


Figure 13. Compute Vehicle Plant Dynamics

MODE: AS2

A22 Text: The stick inputs (1C2), rudder inputs (1C2), and vehicle configuration (1C1) are used by (1) to compute roll, pitch, and other associated model variables (101). The throttle inputs (2C2) and engine configuration (2C1) are used by (2) to compute the engine thrust, RPM, and other associated model variables (201). When vehicle initialization inputs (3C1) are received by (3) the vehicle position, angles, and rates are initialized. Model variables (4C2, 4C3), vehicle initialization variables (4C1), and environment (4C4) (such as windspeed) are used by (4) to compute the aircraft flight path and to update the vehicle state variables (401).



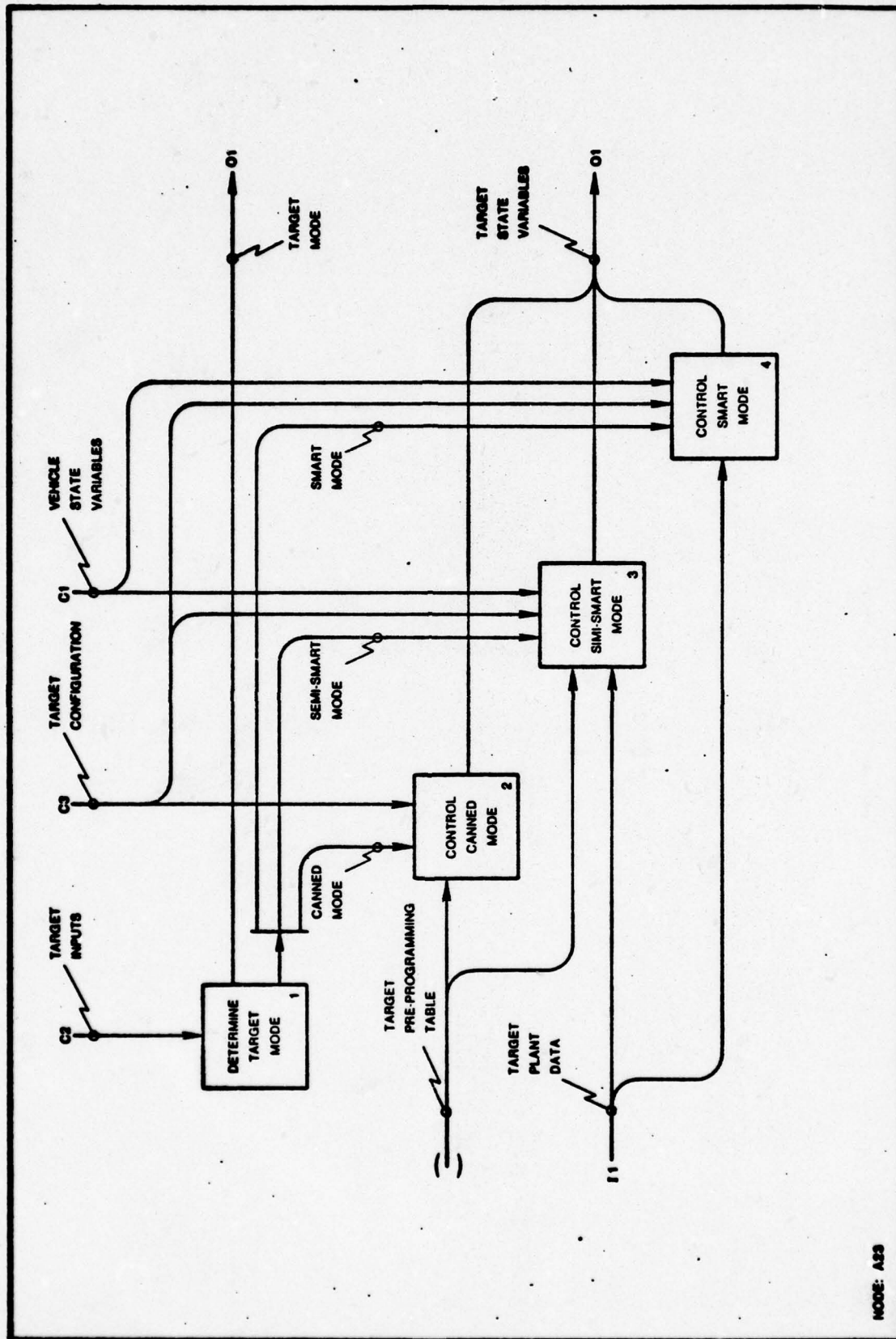
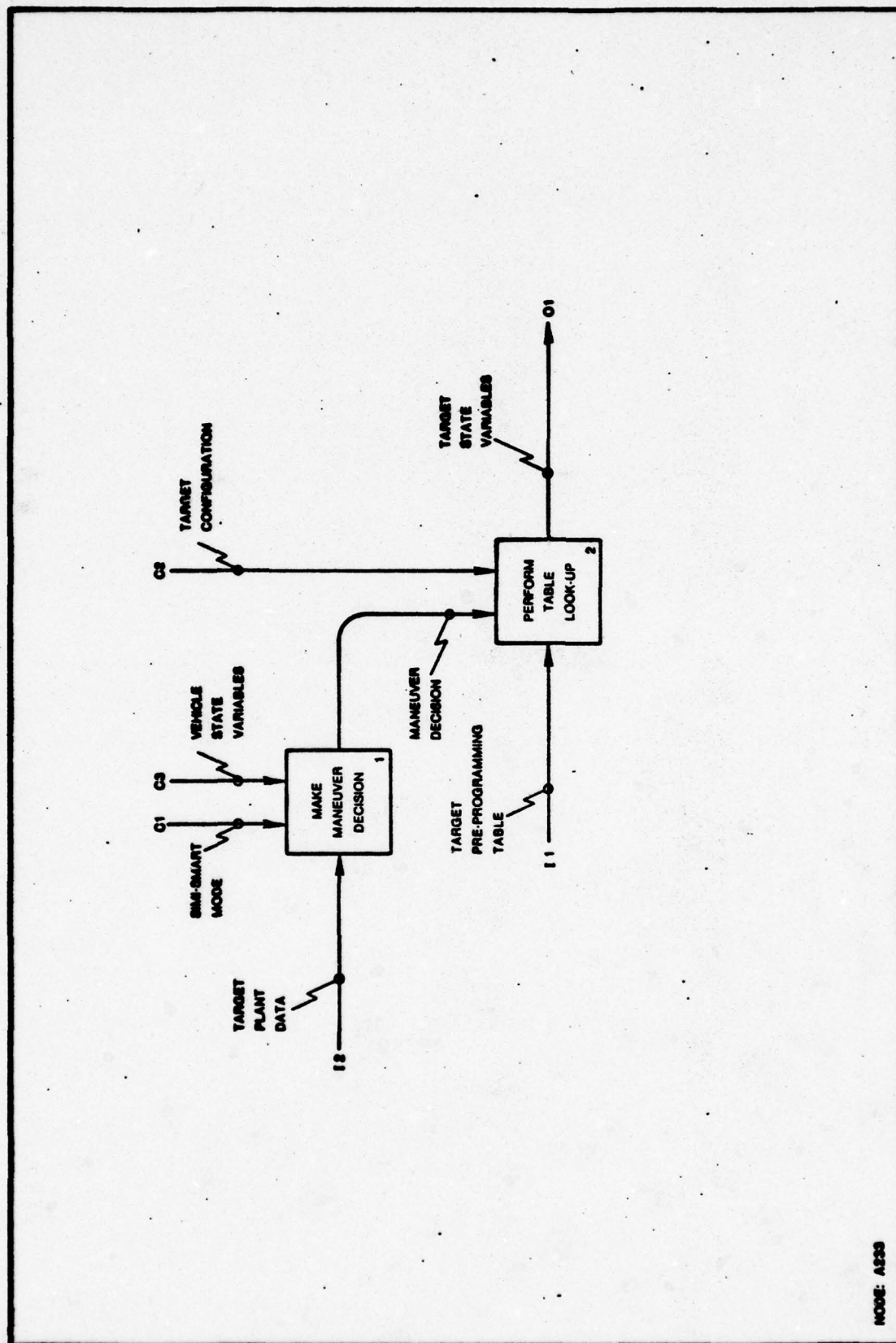


Figure 14. Compute Target Plant Dynamics

A23 Text: The target mode (101) is determined by (1) from the target inputs (1C1). The target mode (102, 103, 104) activates the appropriate activity (2, 3, 4), which controls and generates the target flight path (201, 301, 401). The target configuration (2C2) is used by (2) to control a table lookup for initializing and maneuvering a canned target. The semi-smart target is initialized and maneuvered by (3). The flight path (301) of the semi-smart target is obtained from a preprogrammed table, based upon the attack vehicle maneuvers (3C3). The smart target maneuvers (401) are generated by a computer algorithm; this algorithm is controlled by the vehicle state variables (4C3) and the target configuration (4C2).



MODE: A233

Figure 15. Control Semi-Smart Mode



A233 Text: Using a specified algorithm and knowing the vehicle state variables, (1) decides what evasive maneuver to utilize. This maneuver decision is used by (2) to control a preprogrammed table look-up, to generate the target variables (201).

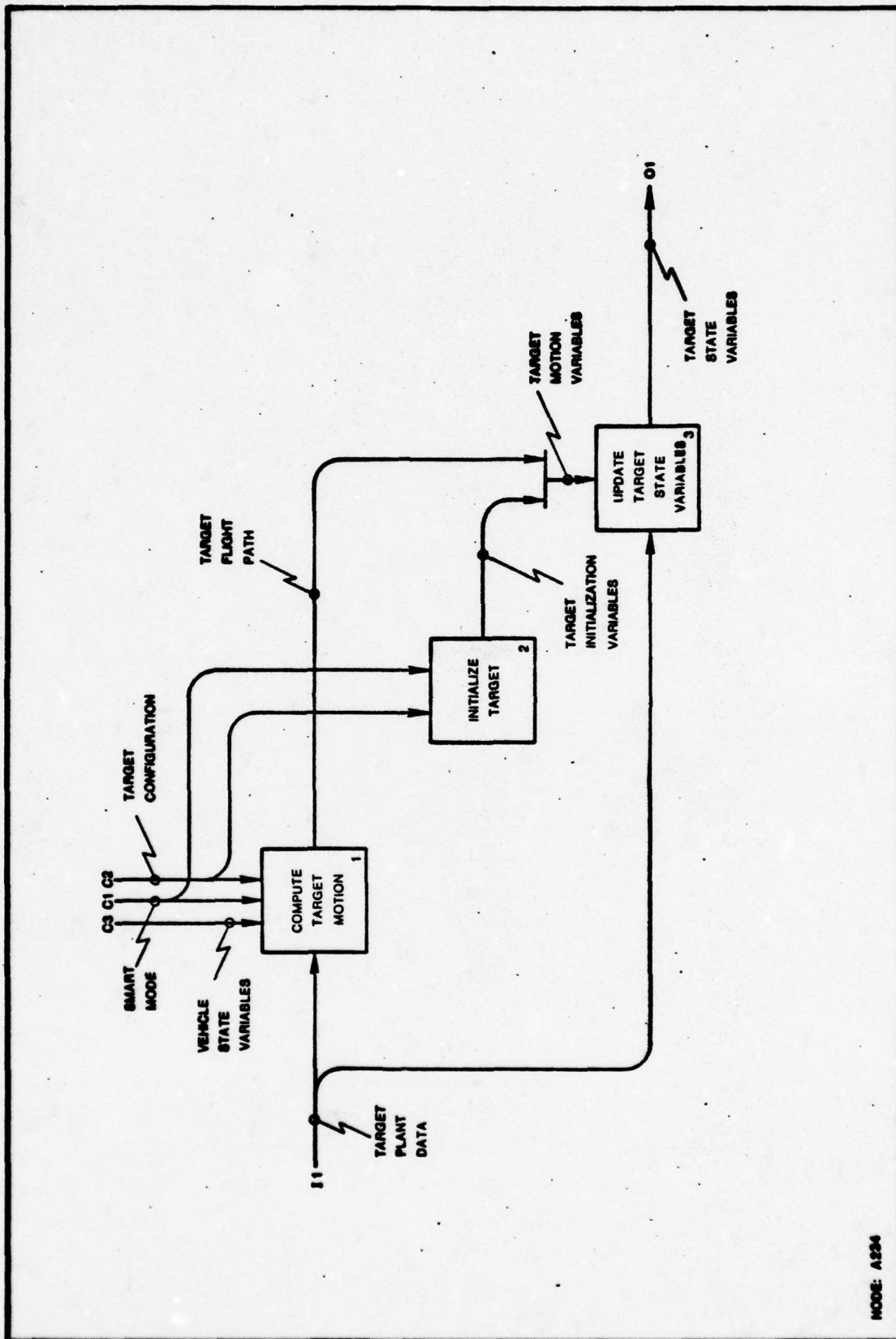
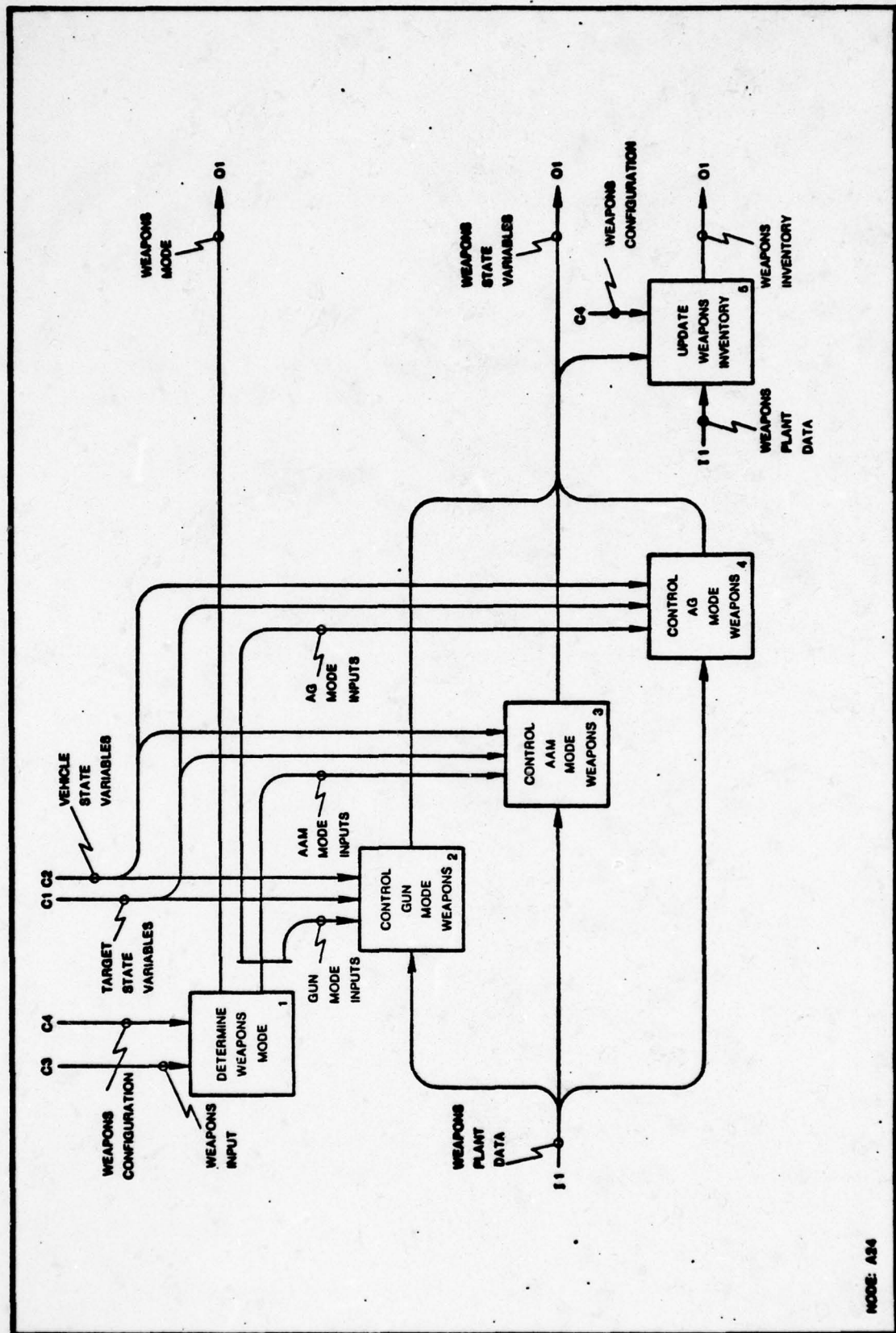


Figure 16. Control Smart Mode

MODE: A224

A234 Text: The target configuration (1C3) and vehicle state variables (1C1) are used by (1) to control the computer algorithm, for computing the target flight path (101). The smart mode initialization (2C2) is used to initialize the target position, angles, and rates (201). The target state variables (311) are updated by (3) from the target motion variables (3C1).





MODE: AS4

Figure 17. Compute Weapons Plant Dynamics

A24 Text: The weapons mode (101) is determined by (1) from the weapons inputs (101) and weapons configuration (102). The weapons mode inputs (102, 103, 104) are passed to the appropriate activity to control the weapons mode and to generate the weapons state variables (201, 301, 401), such as the type and the number of weapons fired.

These activities (2, 3, 4) prepare the weapons for release, compute the lead angle, and simulate the release of the weapons when desired. The weapons inventory (511) is then updated by (5), according to the number of weapons released (501).

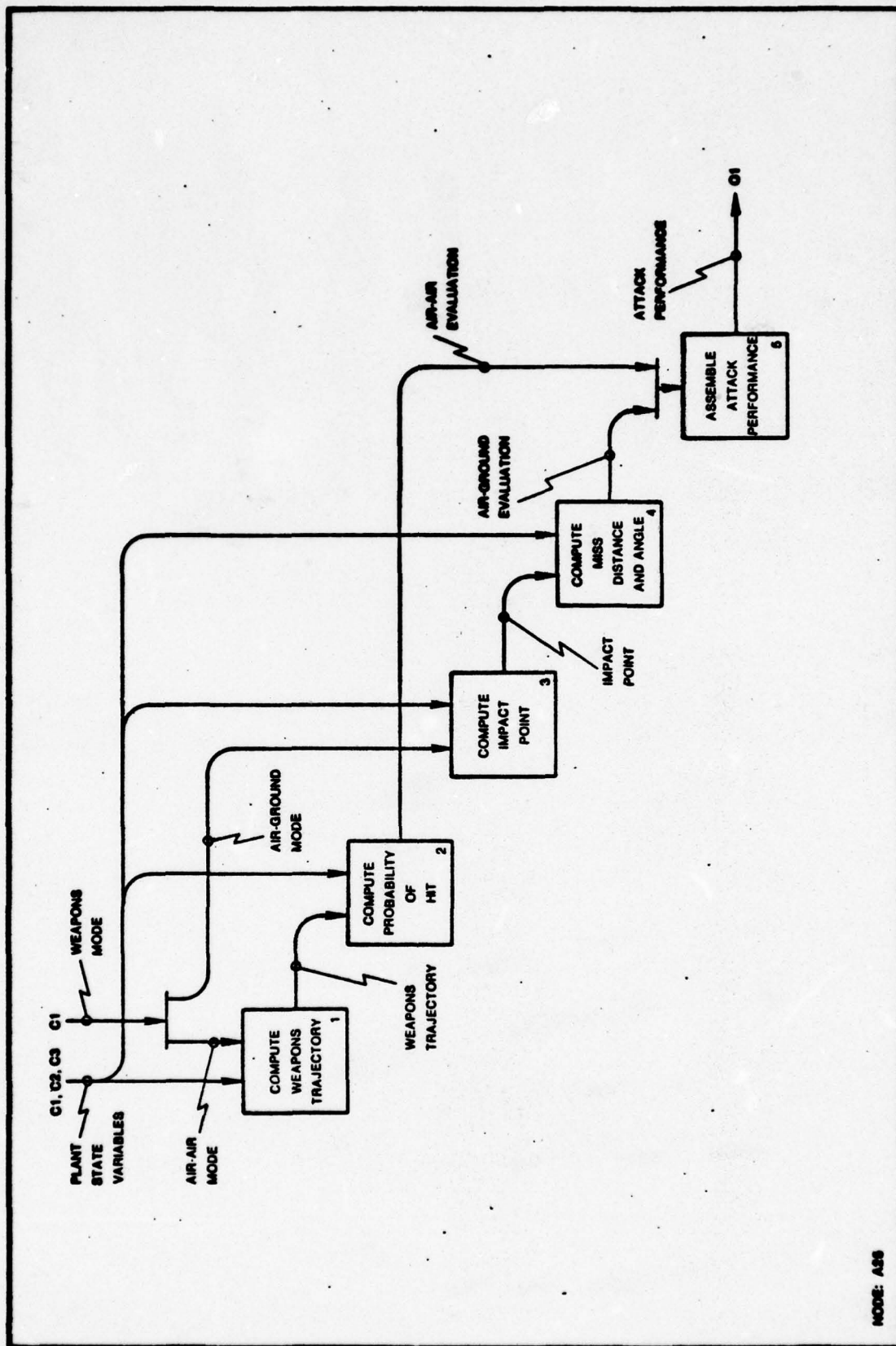


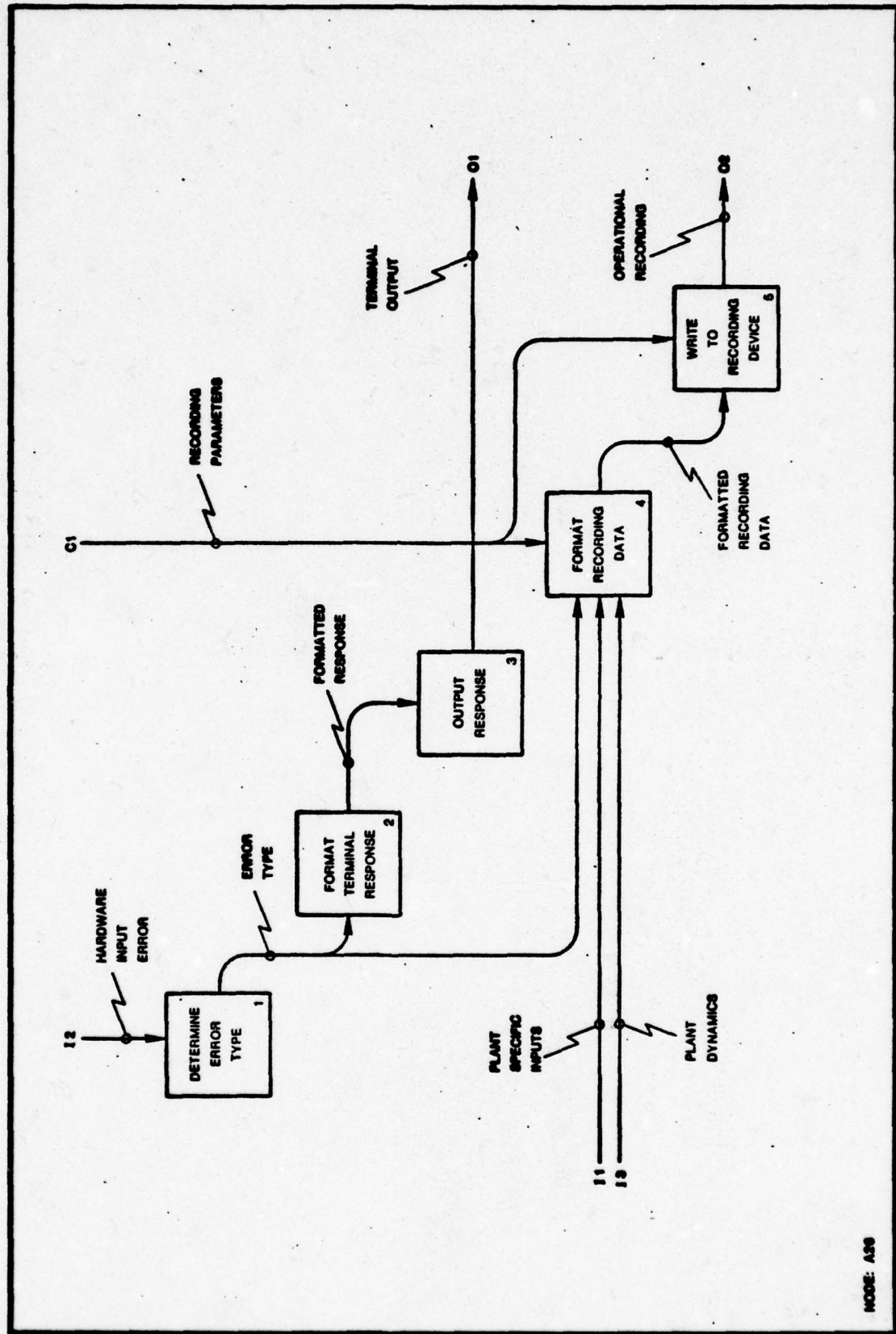
Figure 18. Compute Attack Performance

MODE: ASS



425 Text: When in the air-air mode (1C2), the weapon trajectory (101) is computed by (1). This computation is controlled by the appropriate plant state variables (1C1), such as the position, angle, and rates of the aircraft and the weapon drag coefficients. The weapon trajectory path (2C1) is compared to the flight path of the target (2C2), and the probability of a hit (201) is computed.

When in the air-ground mode (3C1), (3) uses known characteristics of the weapon motion (3C2) and the release point (3C2) to compute the weapon impact point (301). (4) compares the predicted impact point (4C1) with the known target location (4C2) to compute the miss distance and angle (401). These evaluation results (5C1, 5C2) are assembled by (5) into an attack performance report (501).



MODE: A39

Figure 19. Output Operational Recording

**A26 Text:** The hardware error type (101) is determined by (1) from the hardware input error messages (101). The error type (211) is formatted by (2) and output to the user terminal by (3). The error type (411), plant specific inputs (412), and plant dynamics (413) are formatted by (4) into proper recording format according to the specifications of the recording parameters (4C1). The formatted recording data (511) are written to a recording device by (5), producing an operational recording (501).



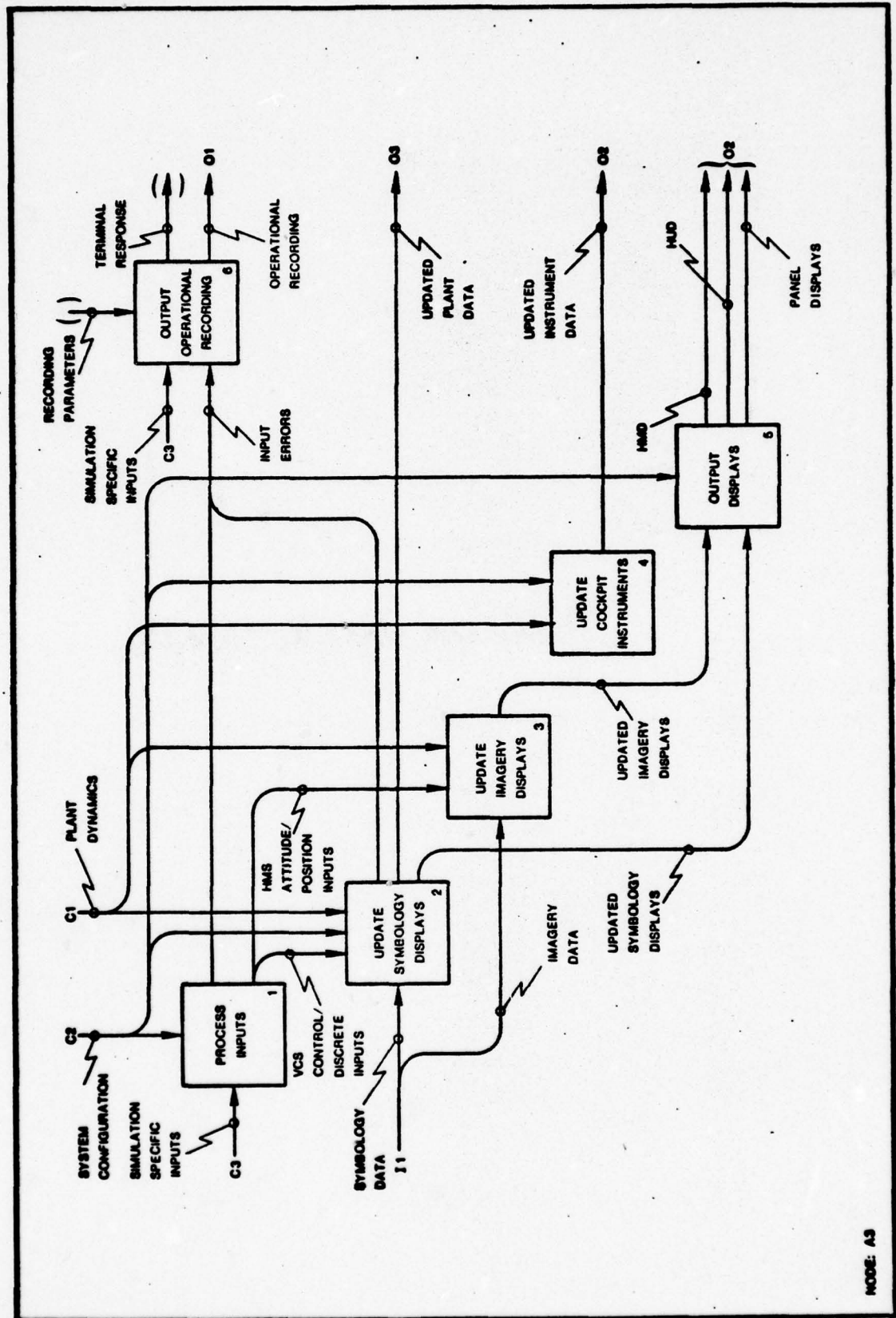


Figure 20. Update Aircraft Displays

MODE: A3

**A3 Text:** The simulation specific inputs (VCS control, discrete, HMS attitude, and position inputs) are received by 'Process Inputs' (1). If there are no hardware input errors, the inputs are formatted and used to control the activities (2, 3). The hardware input errors (101) are recorded as part of the system history and an error message (601) is output to the user by (6).

The VCS control and discrete inputs (2C1), such as the table and menu inputs, are checked for validity by (2). The valid inputs are used to update the symbology displays (203) and plant data (202). The invalid inputs (201) are recorded as part of the system history, and an error message is output to the user by (6).

The symbology displays are the table and menu displays, the helmet mounted displays (HMD), the head-up display (HUD), and the panel displays. The helmet mounted sight (HMS) attitude and position inputs (3C1), weapons state variables (3C2), and target state variables (3C2) are used by (3) to update the imagery displays (301). The imagery displays consist of the target and the weapon envelope display.

The cockpit instruments are updated by (4) from the respective vehicle state variables (4C2). These instruments include the fuel gauge, magnetic compass, airspeed indicator, altimeter, and power indicator. The updated displays (511, 512) are formatted and are output to the appropriate hardware display device by (5). The simulation specific inputs (611) are recorded by (6) as part of the system history.

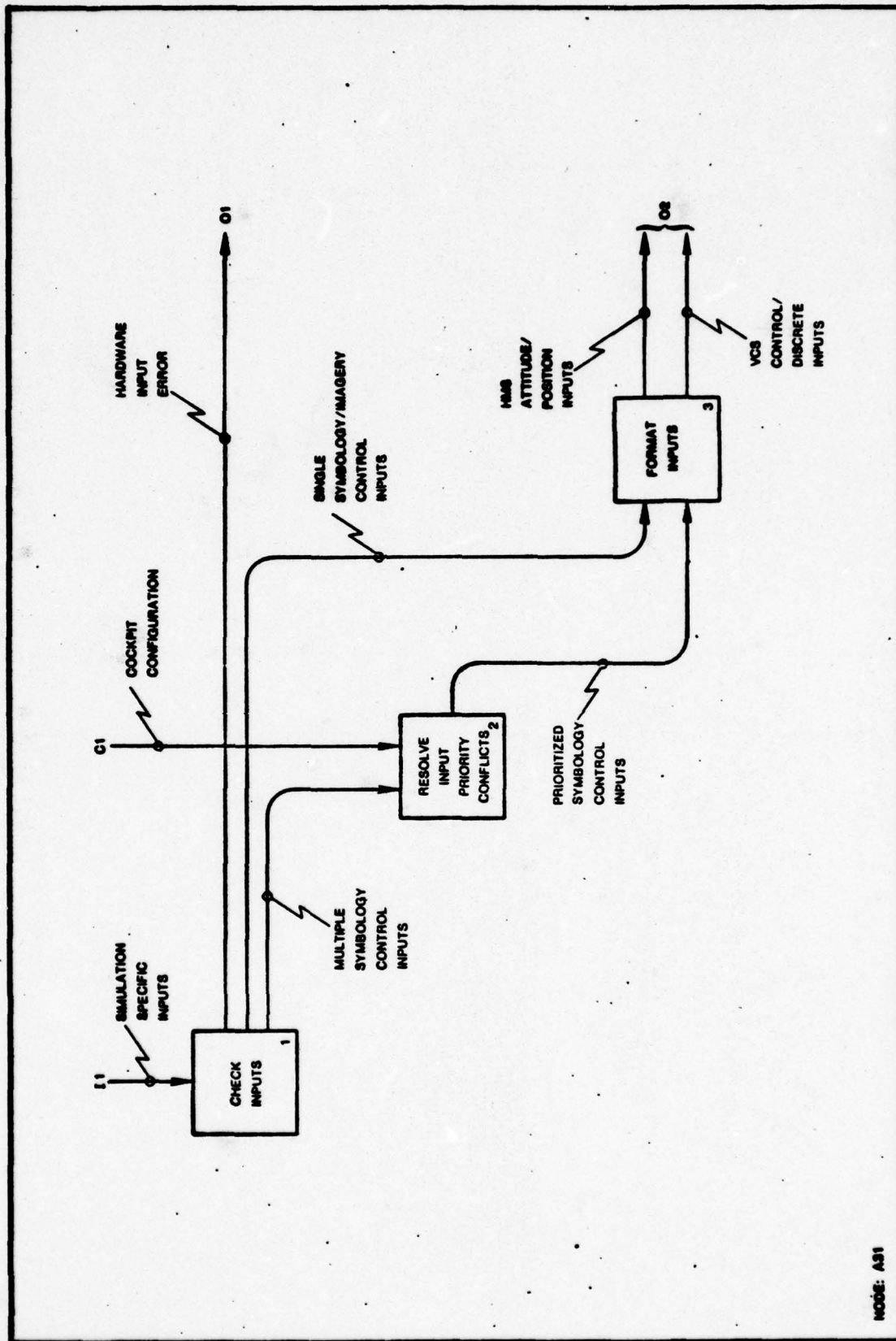


Figure 21. Process Inputs

MODE: ASI



A31 Text: 'Check Inputs' (1) receives simulation specific inputs (1C1) and checks them for validity. If a hardware error is found, a corresponding hardware input error message (101) is generated. The valid single symbology and imagery control inputs are formatted by (3); valid multiple symbology control inputs are passed to (2). With a dual cockpit configuration, it is possible to receive the same type of symbology control inputs from both cockpits. These multiple input conflicts (2C1) are resolved by (2) and the prioritized symbology control inputs (201) are formatted by (3).

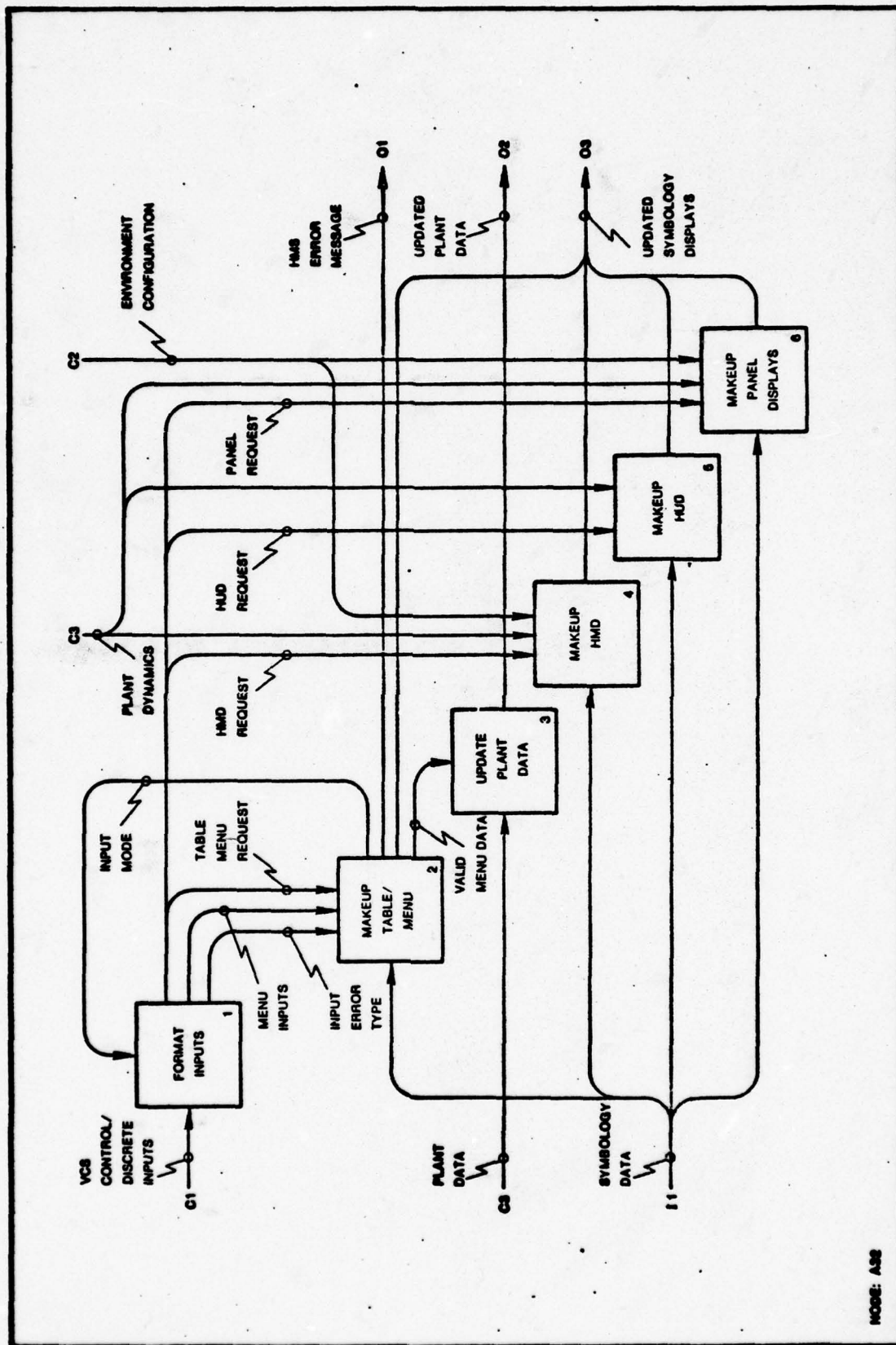


Figure 22. Update Symbology Displays

MODE: ASS

**A32 Text:** 'Format Inputs' (1) checks the VCS control and discrete inputs (111) for validity and type. If an invalid input is found, the appropriate error message (103) is generated and displayed to the user by (2), and is output for operational recording. Valid inputs (111) are formatted, into either menu data inputs (102) or display requests (101) for system use. The input mode (201) is determined by 'Makeup Table/Menu' (2) and fed back to (1) as a control (101) for determining the state of the table or menu. The valid formatted request (101) and inputs (102) are passed to the appropriate activity, where the corresponding symbology displays (202, 401, 501, 601) are made or updated. The plant data (311) is updated by (3) from the valid menu data inputs (301).



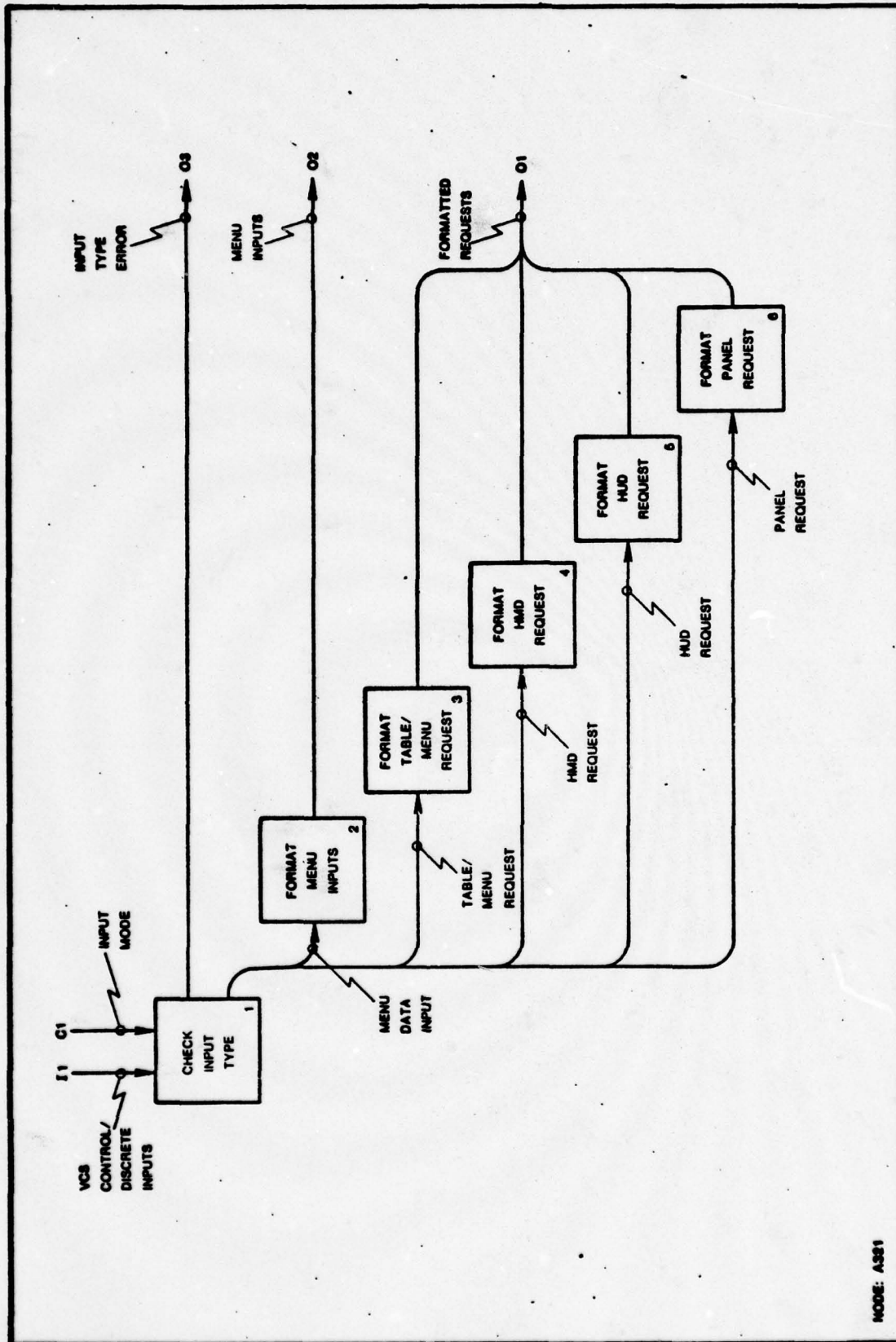
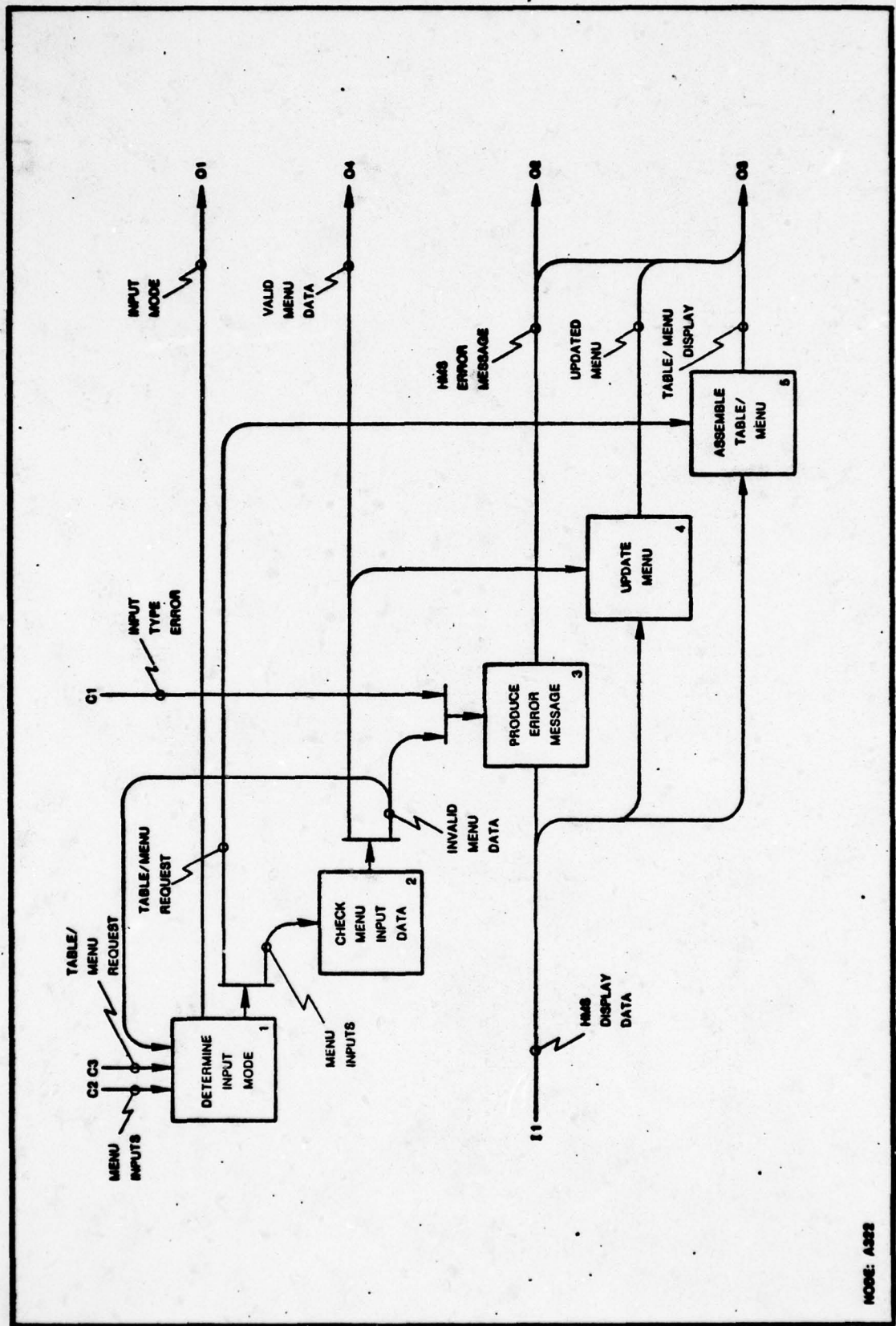


Figure 23. Format Inputs

MODE: A331

A321 Text: 'Check Input Type' (1) checks the type and validity of the VCS control and the discrete inputs (1C1). For invalid type inputs, (1) produces an input error message (101). The valid inputs are passed to the corresponding activity to be formatted for system use. The input mode (1C2) is used by (1) to determine the state of the table or menu. This is necessary since the menu data inputs (211) and the table and menu requests (311) are sequence dependent.



MODE: A322

Figure 24. Makeup Table/Menu



**A322 Text:** The menu data inputs (1C1), table and menu requests (1C2), and, when appropriate, the invalid data message (1C3) are used by (1) to determine and/or change the table and menu input mode. The invalid data message (1C3) keeps the input mode from changing when menu inputs (1C1) are invalid. The menu data inputs (1C3) are checked by (2) for validity. If the data is invalid (2C2), or if there is an input type error message (3C2), the appropriate helmet mounted sight (HMS) error message (3C1) is produced by (3). Valid data (2C1) is output by (2) for updating the plant data and is used by (4) to update the menu display (4C1). The requested table and menu display (5C1) is assembled from HMS display data (5I1), according to the table and menu requests (5C1).

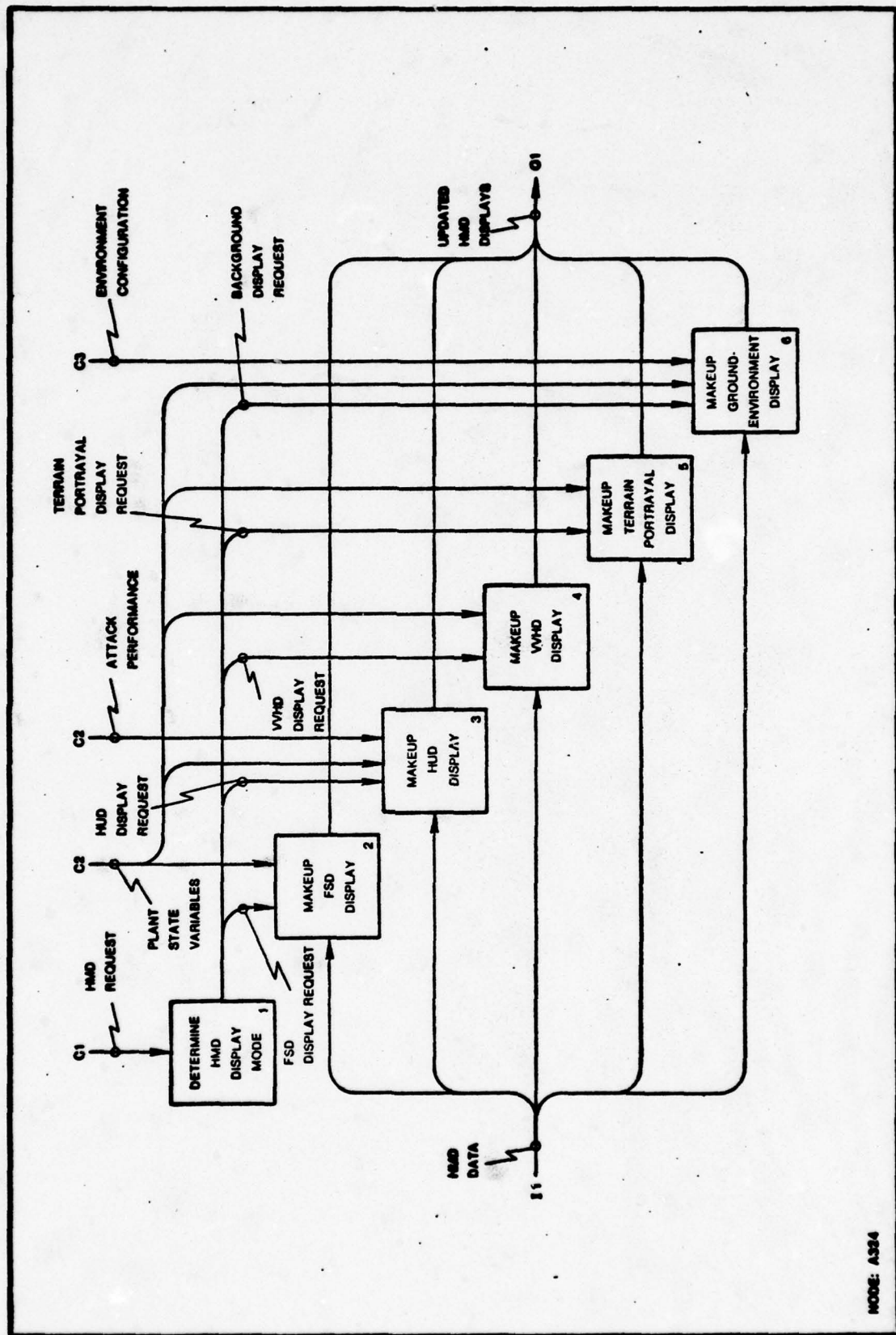


Figure 25. Makeup HMD

A324 Text: The HMD request (101) is used by (1) to determine the HMD display mode (101). The mode (201, 301, 401, 501, 601) activates the appropriate activity and the requested HMD display (201, 301, 401, 501, 601) is generated. A description of these displays is found in Chapter II.



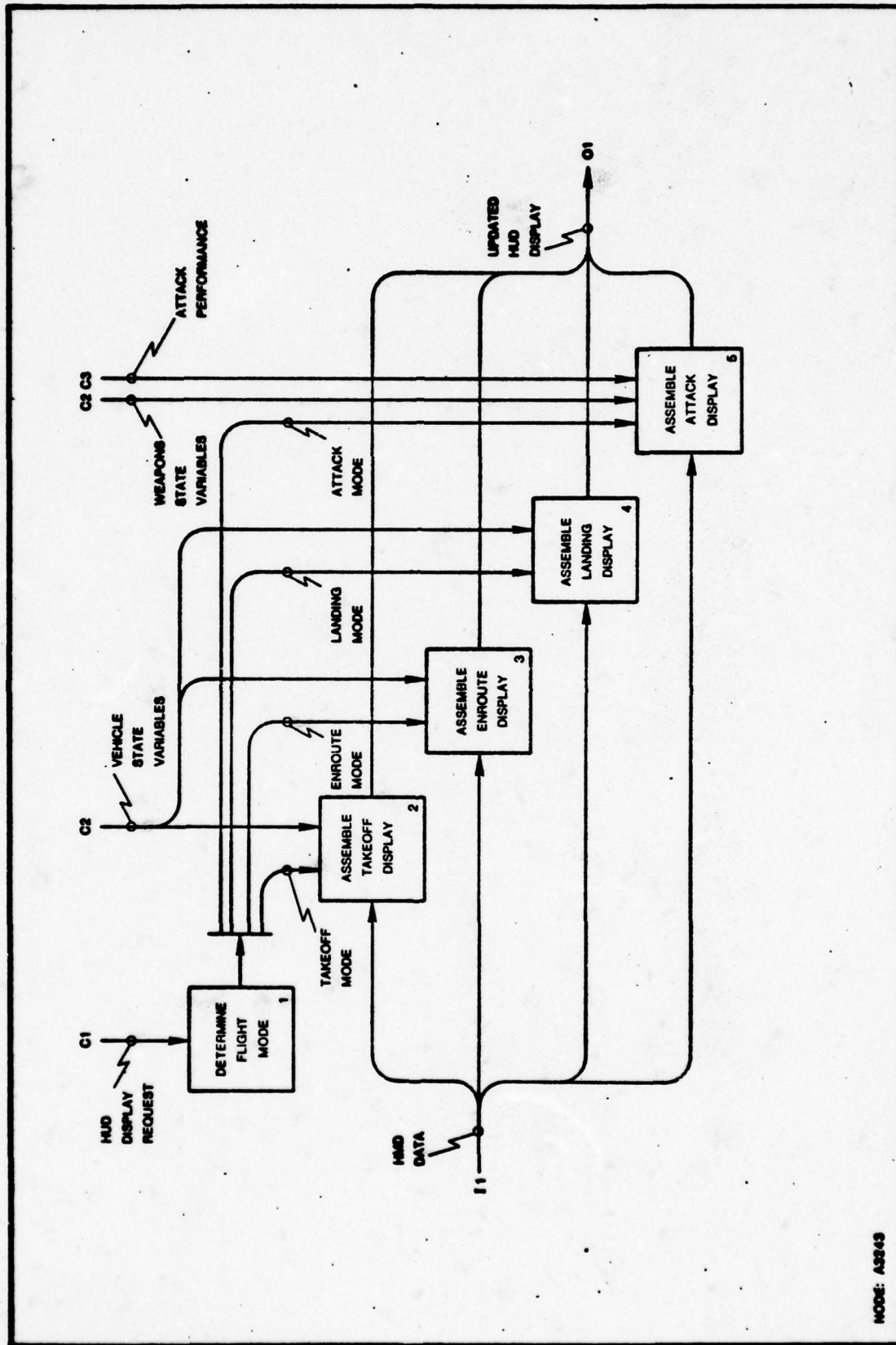


Figure 26. Makeup HUD Display (HMD)

MODE: A3843

**A3243 Text:** The flight mode (101, 102, 103, 104) is determined by (1) from the HUD display request (101). This mode activates the appropriate activity, and the proper HUD flight display (201, 301, 401, 501) is generated. The takeoff, enroute, and landing displays (201, 301, 401) contain information about the aircraft altitude, heading, airspeed, pitch, roll, and horizons. The attack display (501) contains weapon sights (501), associated lead angles (501), weapons available (501), weapons fired (501), and number of hits (502).

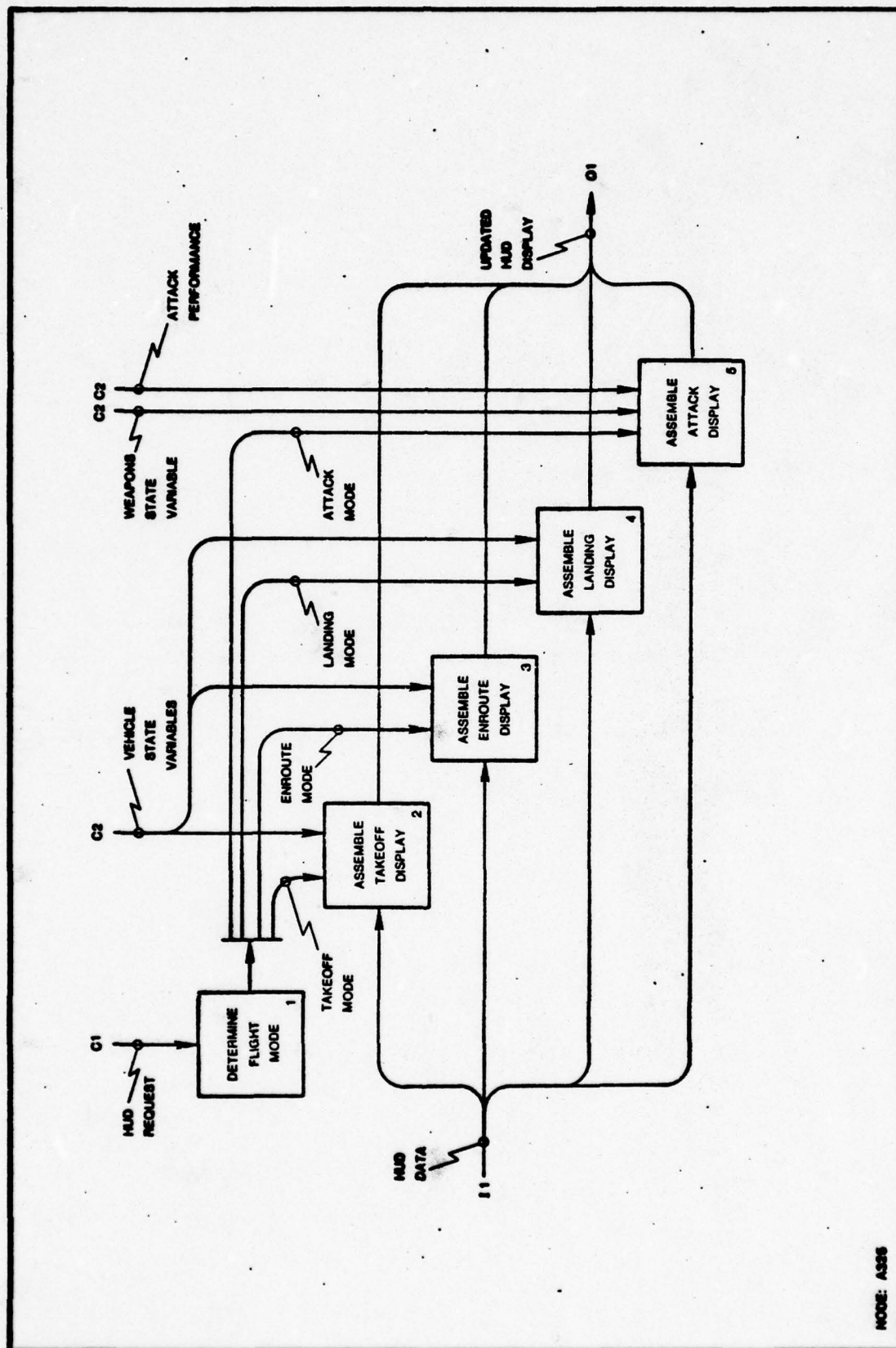


Figure 27. Makeup HUD



A325 Text: The flight mode (101, 102, 103, 104) is determined by (1) from the HUD request (101). The mode activates the appropriate activity, and the proper HUD flight display (201, 301, 401, 501) is generated. The takeoff, enroute, and landing displays (201, 301, 401) contain information about the aircraft altitude, heading, airspeed, pitch, roll, and horizons. The attack display (501) contains weapon sights (5C1), associated lead angles (5C1), weapons available (5C1), weapons fired (5C1), and number of hits (5C2).

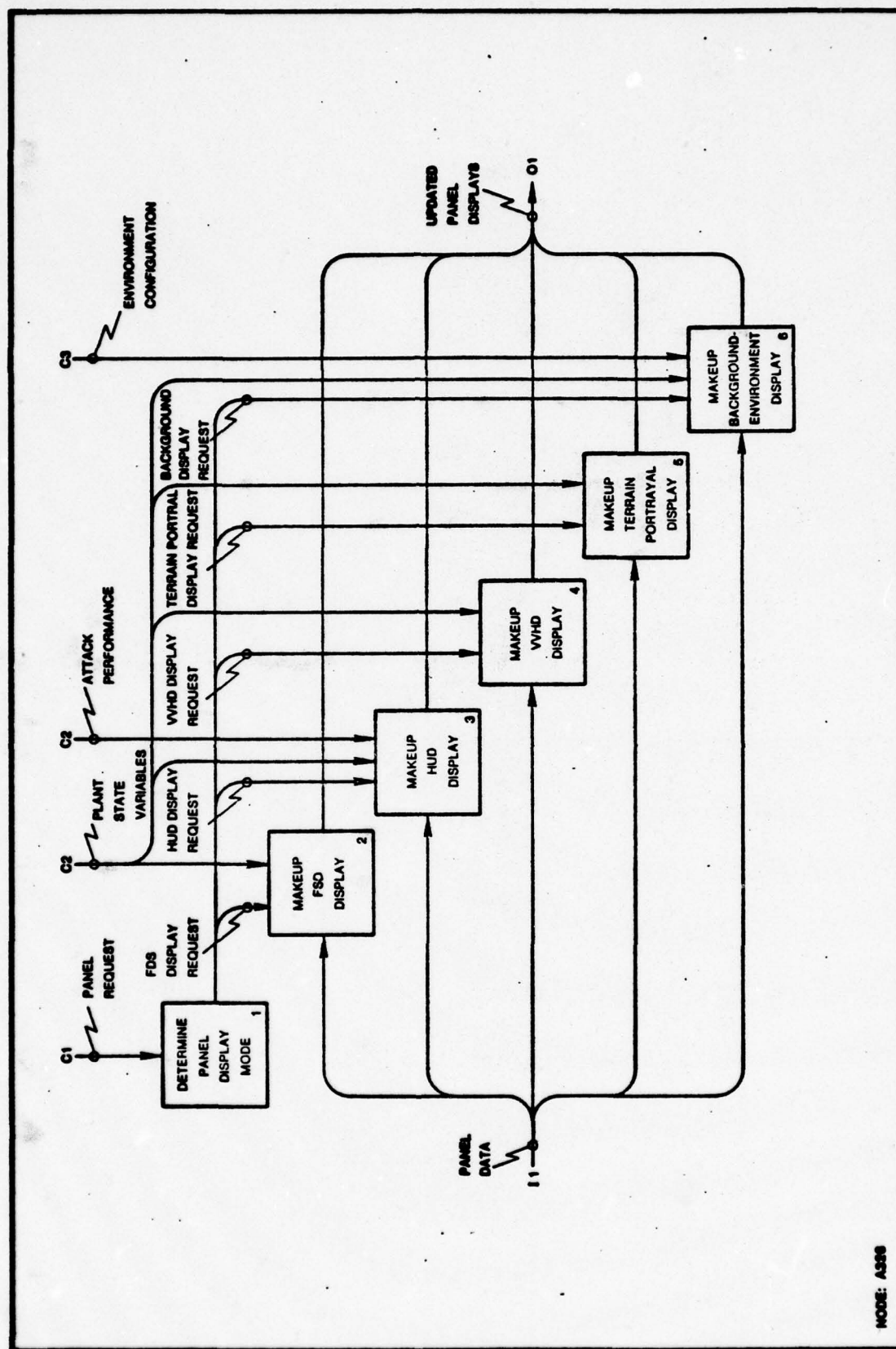
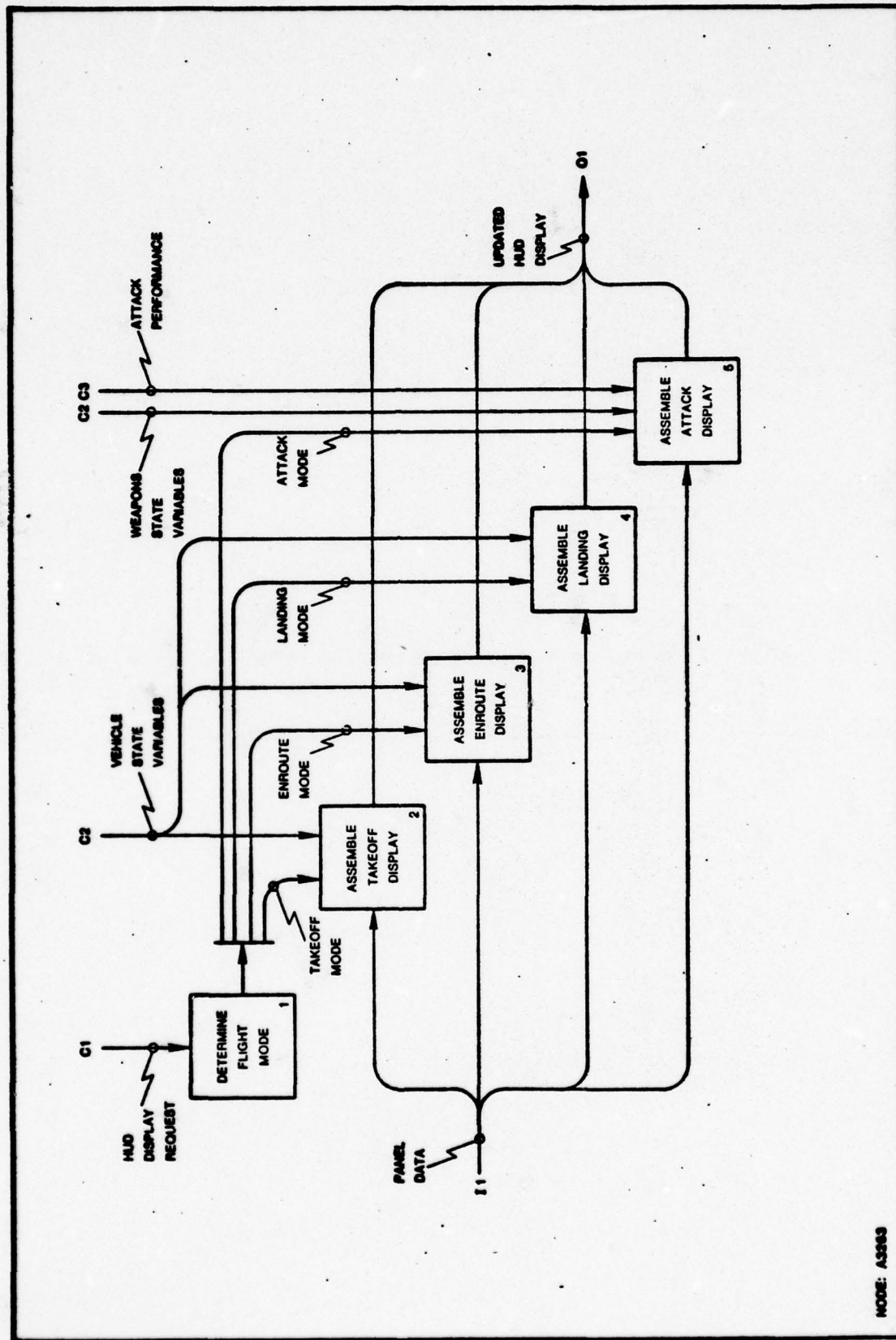


Figure 28. Makeup Panel Displays

MODE: A398

A326 Text: The panel request (101) is used by (1) to determine the panel display mode (101). This mode (201, 301, 401, 501, 601) activates the appropriate activity and the requested panel display (201, 301, 401, 501, 601) is generated. A description of these displays is found in Chapter II.





MODE: A3303

Figure 29. Makeup HUD Display (Panel)

**A3263 Text:** The flight mode (101, 102, 103, 104) is determined by (1) from the HUD display request (101). This mode activates the appropriate activity, and the proper HUD flight display (201, 301, 401, 501) is generated. The takeoff, enroute, and landing displays (201, 301, 401) contain information about the aircraft altitude, heading, airspeed, pitch, roll, and horizons. The attack display (501) contains weapon sights (5C1), associated lead angles (5C1), weapons available (5C1), weapons fired (5C1), and number of hits (5C2).

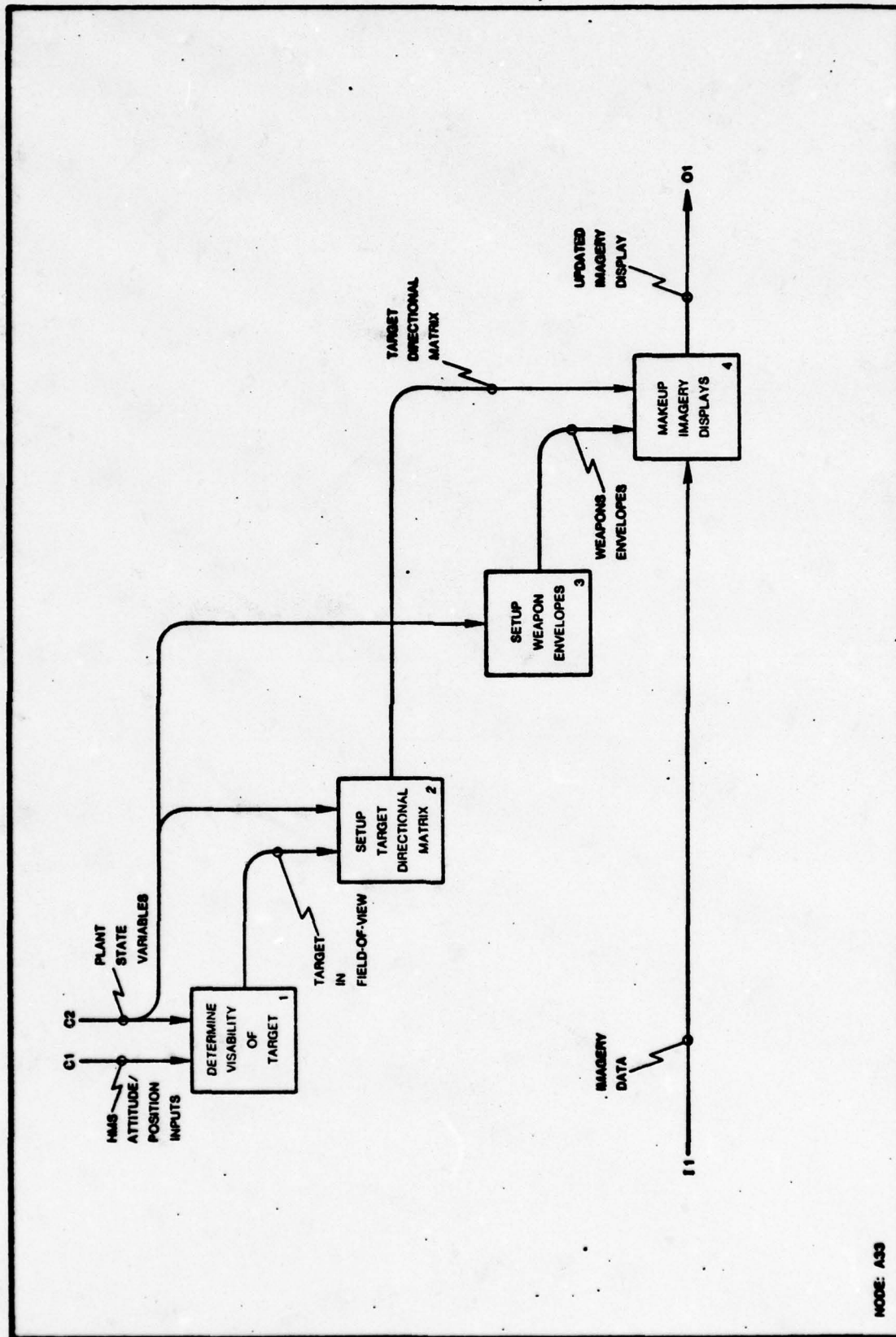


Figure 30. Update Imagery Displays

MODE: A33



A33 Text: The helmet mounted sight (HMS) attitude and position (1C1), vehicle state variables (1C2), target state variables (1C2), and the environment configuration (1C2), such as the visibility factor, are used by (1) to determine if a target is in the field-of-view (101). When the target is in the field-of-view (2C1), the target state variables (2C2), such as heading, altitude, and velocity, are used to set up the target directional matrix (201). The weapons state variables (3C1), such as the weapon effective range, are used by (3) to set up the weapon envelope (301). The weapon envelope (4C1) and the target directional matrix (4C2) are used by (4) to update and/or makeup the imagery display (401).

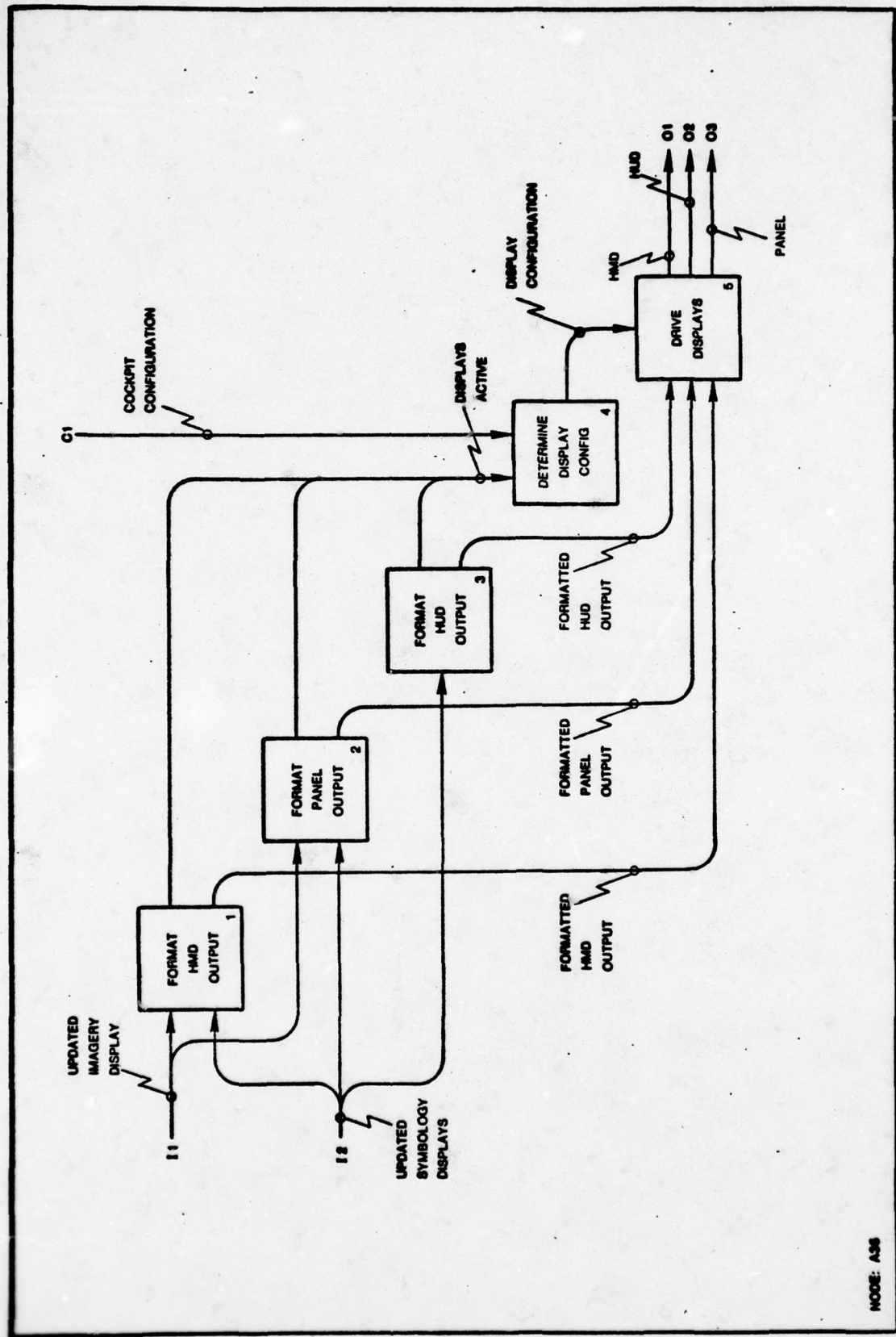


Figure 31. Output Displays

**A35 Text:** The updated imagery display (111, 211) is received by (1) and (2), and formatted for output (102, 202) on the HMD and panel display devices. The updated symbology display (112, 212, 311) is received by (1), (2), and (3), and is formatted for output (102, 202, 302) on the HMD, panel display, and HUD display devices respectively. The above activities also notify (4) as to the displays that are active (101, 201, 301). The displays active flags (4C1) and the cockpit configuration (4C2) are used by (4) to determine which display devices are to be driven (5).



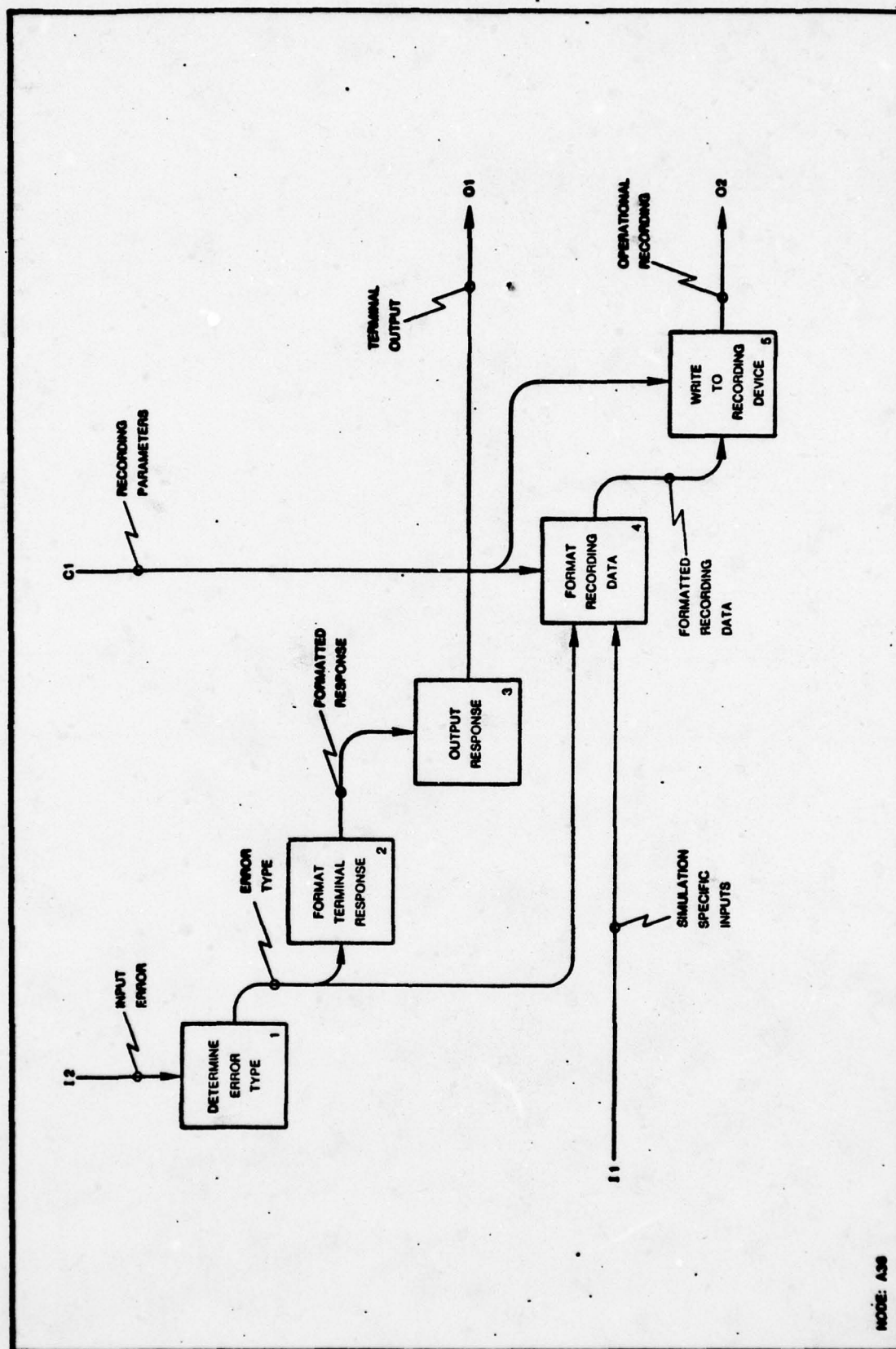


Figure 32. Output Operational Recording

MODE: A38

A36 Text: The error type (101) is determined by (1) from the input error message (101). The error type (211) is formatted by (2) and output to the user terminal by (3). The error type (411) and simulation specific inputs (412) are formatted by (4) into the proper recording format, according to the specifications of the recording parameters (401). The formatted recording data (511) is written to a recording device by (5), producing an operational recording (501).

## Data Model

<u>Node</u>	<u>Title</u>	<u>Page</u>
D-0	Simulator Data	77
D0	Simulator Data	79
D1	Configuration Data	82
D3	Analog/Digital Inputs	84
D4	Plant Data	86
D5	Performance Data	88
D6	Simulation Displays	90



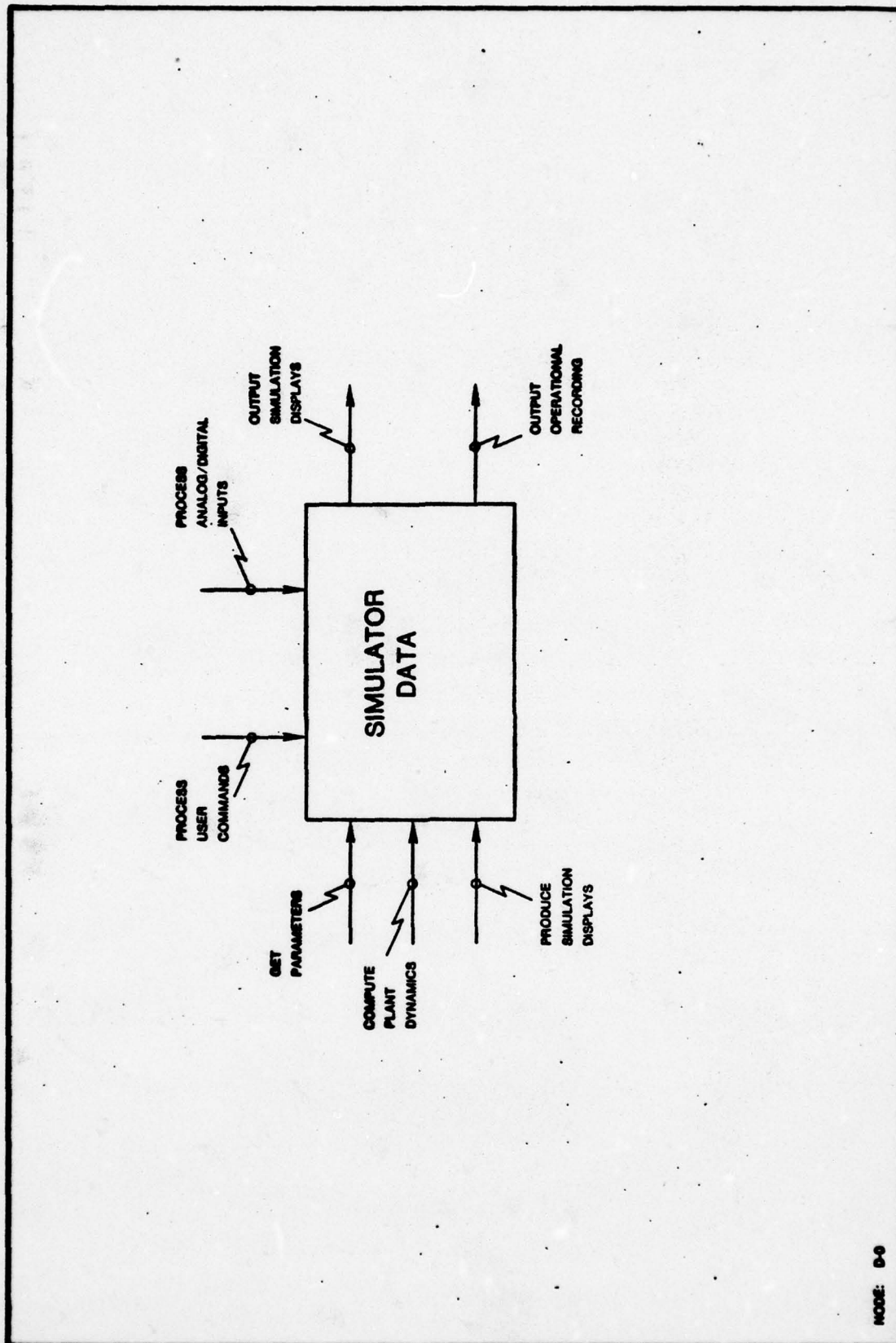


Figure 33. Simulator Data

Purpose and Viewpoint: This data model provides the formal functional specifications for the VCASS from the viewpoint of a system software designer. The system should accept inputs from an aircraft representative vehicle, whose type, threat, and weapons dynamics can be altered; it should provide synthesized out-of-the cockpit instrumentation, visual scenes, and targets.

D-O Text: The activities that create or modify the simulator data are 'Get Parameters' (I1), 'Compute Plant Dynamics' (I2), and 'Produce Simulation Displays' (I3). The creation or modification of simulator data is constrained by the user commands (C1) and the analog and digital inputs (C2). Simulation data is output in the form of simulation displays (O1) and operational recording (O2).

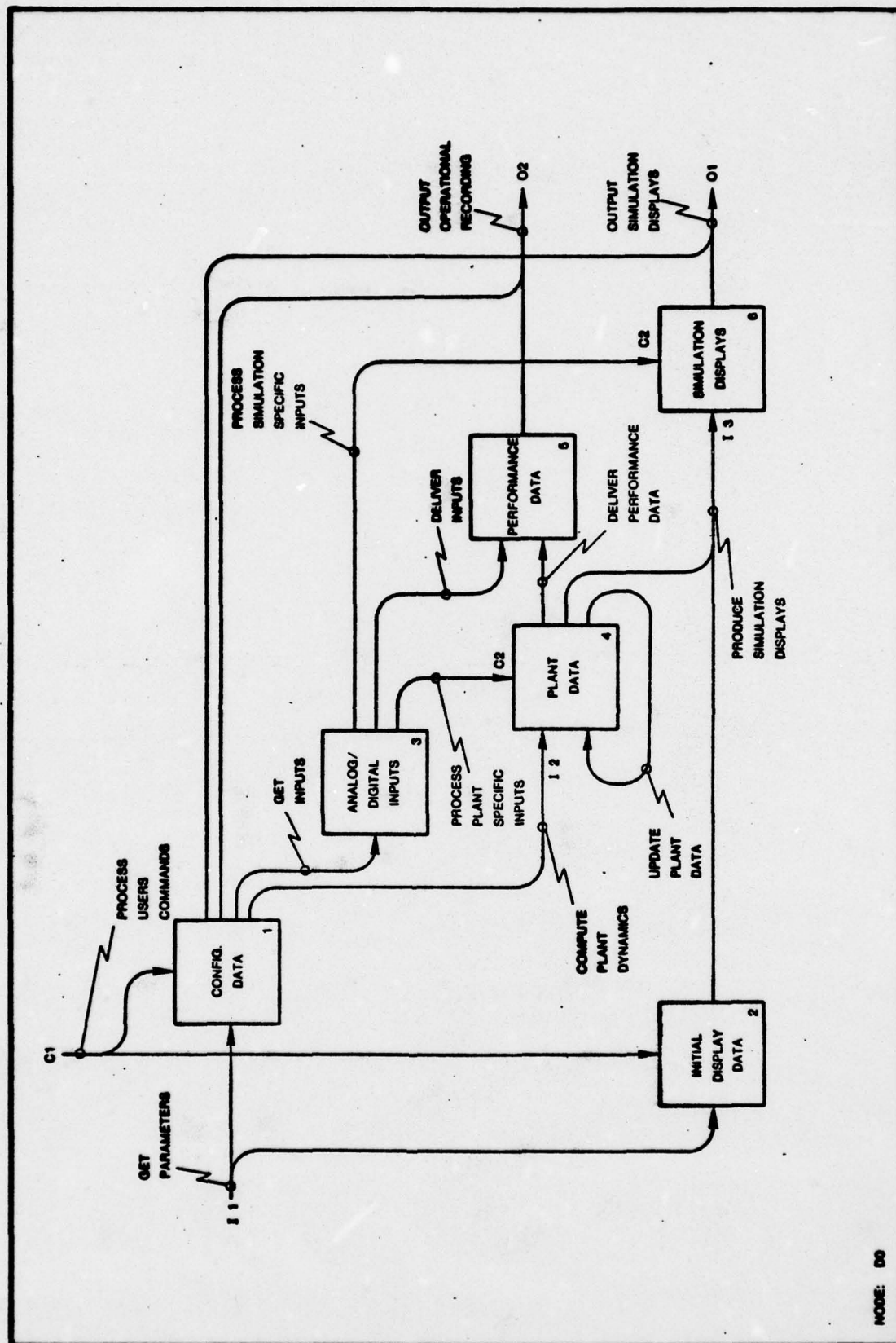


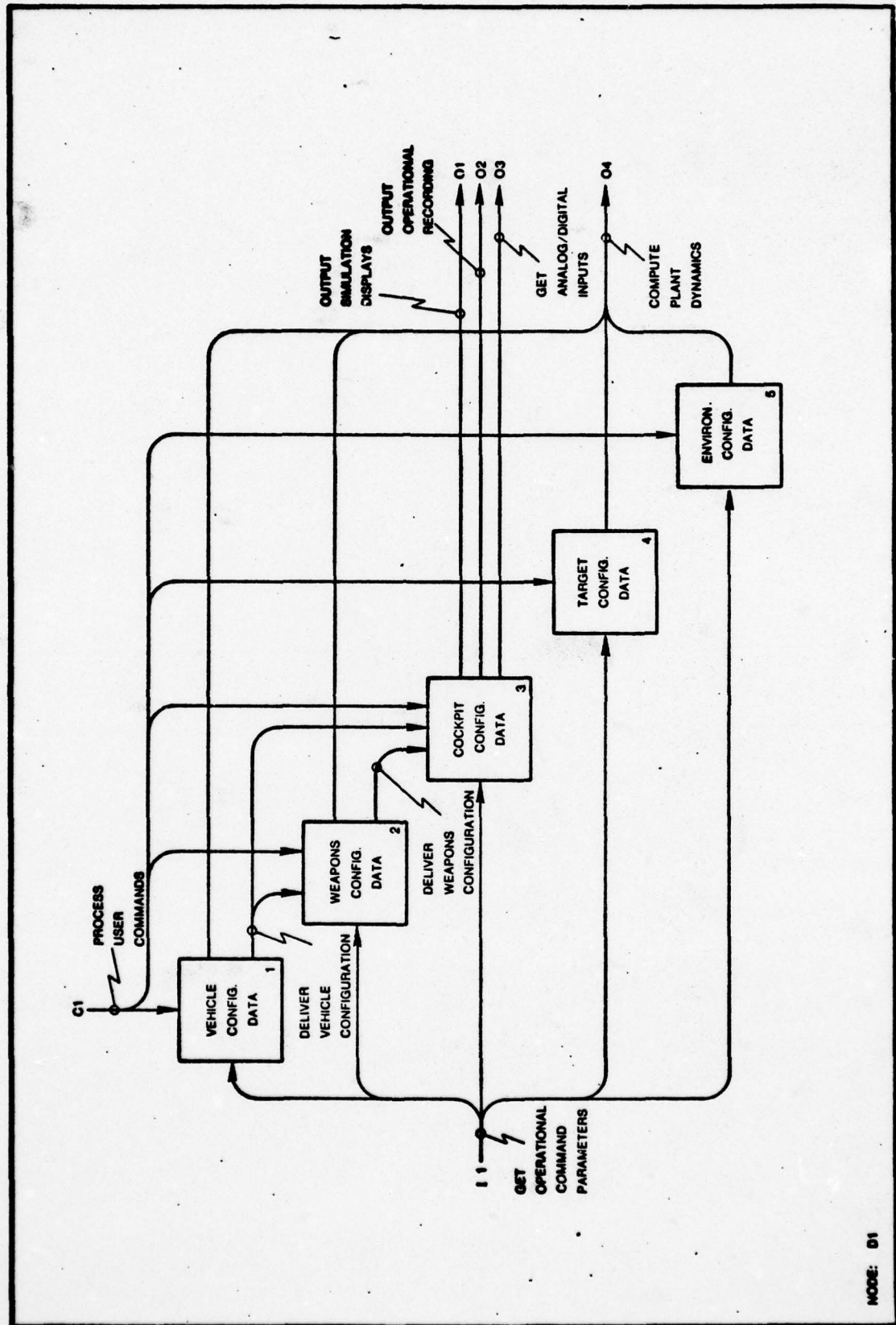
Figure 34. Simulator Data

MODE: DO



DO Text: The configuration data (1) is setup from the input parameters (111) according to the Operational commands (101). Configured are the vehicle, environment, weapons, target, and cockpit. The configuration data (1) is used by the activities which output the simulation displays (101) and records the system history (102). When in a dual cockpit configuration, configuration data (1) controls the resolution of conflicts of inputs (103) from both cockpits. The configuration data (1) also controls the computation of the plant data (vehicle, weapons, target, and attack data) (104). The initial display data (2) is set up from the input parameters (211) according to the Exerciser commands (201). Initial display data (2) contains the formats for the aircraft symbology displays, and is used in the production of the simulation displays (201). The analog and digital-inputs (3) are of two types: plant specific inputs (stick, rudder, target mode, weapon trigger and release) and simulation specific inputs (VCS control inputs, discrete inputs, HMS attitude inputs, and HMS position inputs). The inputs (3) are processed by the respective activity (301, 303) for system utilization. These inputs (3) are also delivered (302) as performance data (5) to be output on the operational recording (501). The processing of the plant specific inputs (401) controls the computation and updating of the plant data (4). Specific portions of the updated plant data (4) are delivered (401) as performance data (5) to be output on the operational recording (501). The plant data (4) is also used in conjunction with the initial data (2) to produce the simulation displays (6). The processing of the simulation specific inputs (601), such as determining the display modes, constrains the production

of the simulation displays (611). These simulation displays are the symbology displays, the imagery displays, and the cockpit instruments.



MODE: 01

Figure 35. Configuration Data



D1 Text: The appropriate data (1, 2, 3, 4, 5) is set-up by the activity 'Process User Commands' (C1). The data (1, 2, 4, 5) is then used to compute the respective plant dynamics (101, 201, 401, 501). The vehicle configuration data (1) contains such things as the aircraft type and associated characteristics. The vehicle configuration (2C1) is used to control the set-up of the weapons configuration data (2) by determining the validity of the weapons parameters entered. The vehicle and weapons configurations (3C1, 3C2) control the set-up of the cockpit configuration data (3) by determining the validity of the cockpit parameters entered. The cockpit configuration data (3) is used to control the routing of the simulation displays (301) and is recorded as part of the system history (302). When in a dual cockpit configuration, the cockpit configuration data (3) is also used to control the resolution of analog and digital input conflicts (303).

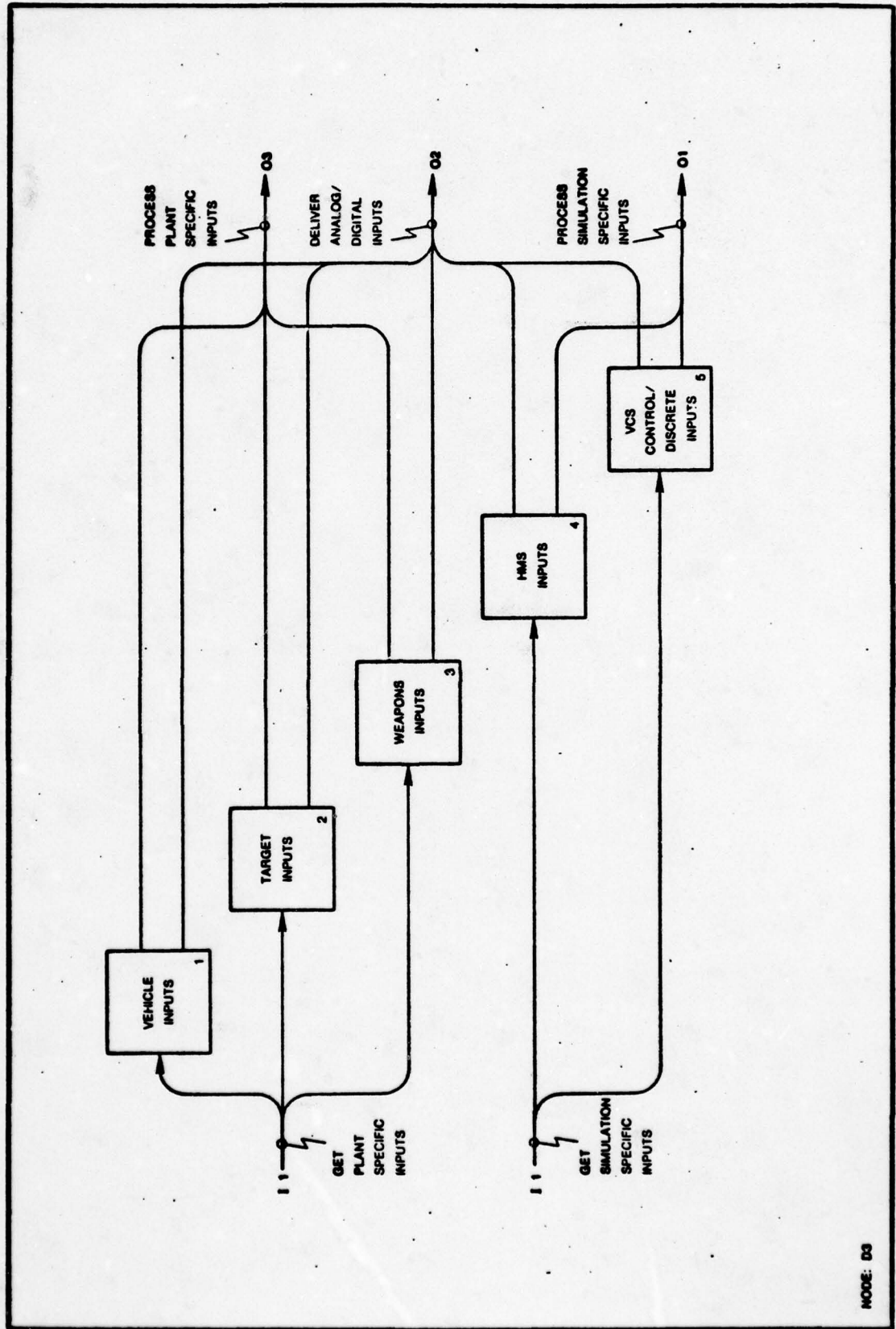


Figure 36. Analog/Digital Inputs

MODE: D3

D3 Text: The vehicle inputs (1) (such as stick, rudder, and throttle), target inputs (2) (such as target mode), and weapon inputs (3) (such as trigger and release) are formatted by (03) for system use. The above data (1, 2, 3), along with the helmet mounted sight (HMS) inputs (4), VCS control inputs (5), and discrete inputs (5) are delivered for operational recording (02). The HMS attitude (4), HMS position (4), VCS control inputs (5), and discrete inputs (5) are formatted by (01) for system use.



AD-A055 226

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 1/4  
SOFTWARE DESIGN FOR A VISUALLY-COUPLED AIRBORNE SYSTEMS SIMULAT--ETC(U)  
MAR 78 W H REEVE, J L STINSON

UNCLASSIFIED

AFIT/GCS/EE/78-6

NL

2 OF 3  
AD  
A055226



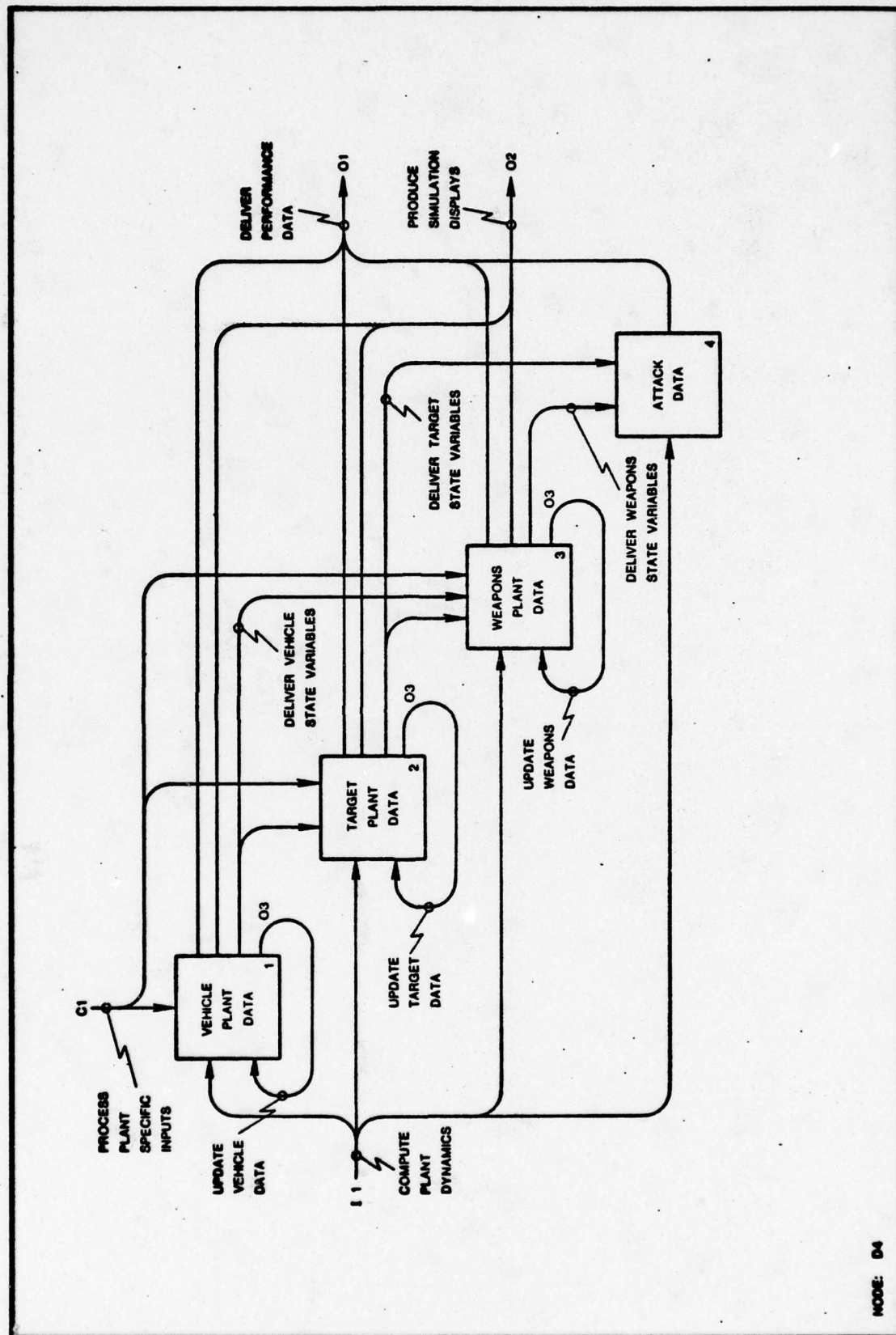


Figure 37. Plant Data

MODE: D4

D4 Text: The plant specific inputs formatted by (C1) are used by the activities which compute (1I1, 2I1, 3I1) and update (1I2, 2I2, 3I2) the corresponding plant data (1, 2, 3).

The vehicle plant data (1), such as heading, velocity, and altitude, is used to control the particular target mode for computing (2I1) and updating (2I2) the target plant data (2). This data (2) contains such information as the target angle, rate, and position of flight. The vehicle plant data (1) and target plant data (2) are used to control the active weapons mode. The weapons plant data (3) contains such information as weapon lead angle and weapon release. The appropriate target plant data (2) and weapons plant data (3) are used to compute (4I1) the attack data (4). The attack data (4) contains such information as the number of hits or miss distance and angle. The vehicle plant data (1), target plant data (2), and the weapons plant data (3) are all used to produce the simulation displays (O2). The above data (1, 2, 3) and the attack data (4) are delivered for recording (O1).



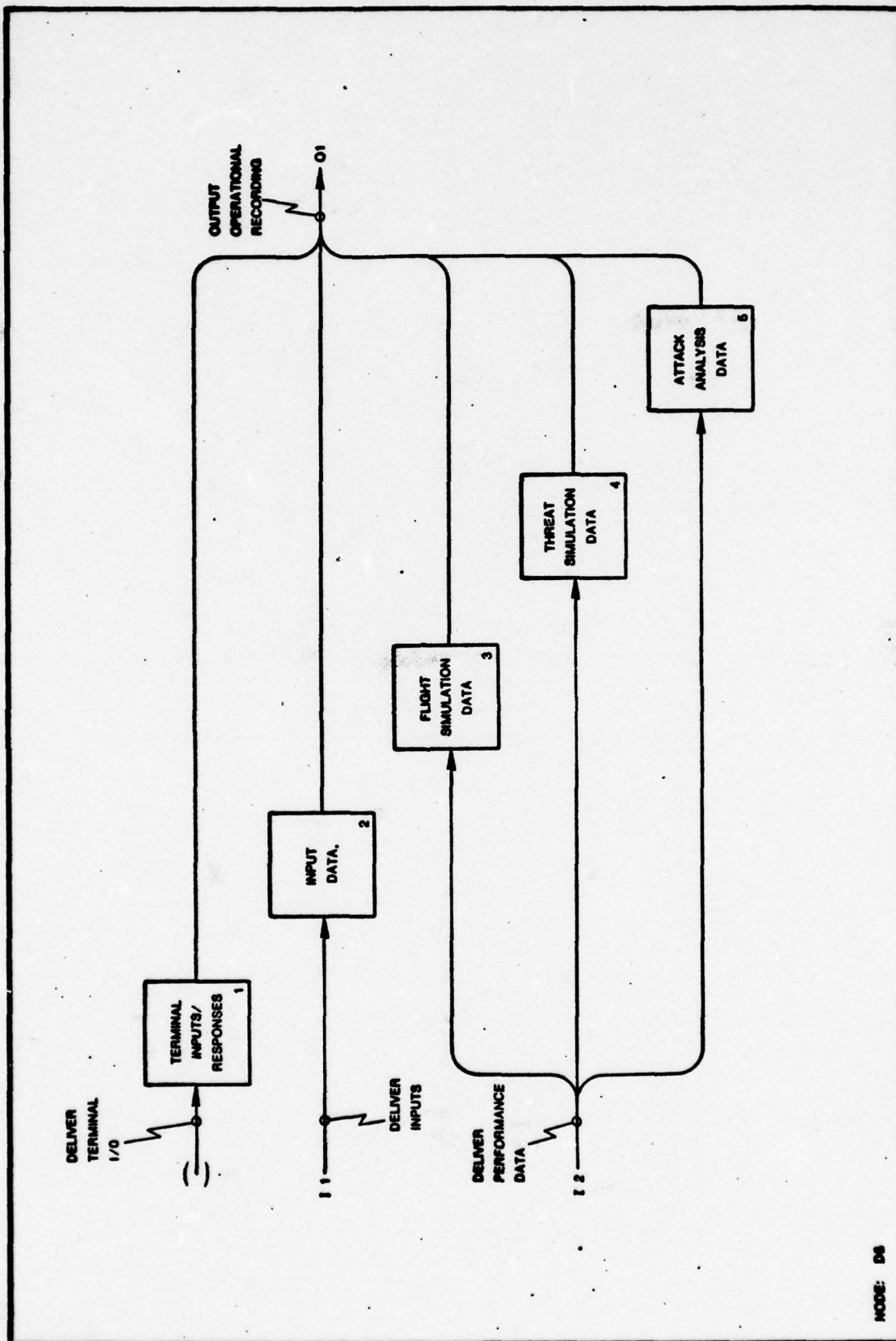


Figure 38. Performance Data

D5 Text: All terminal inputs and responses (1), analog and digital inputs (2), flight situation data (3), threat situation data (4), and attack analysis data (5) are recorded (01) as part of the system history (01).

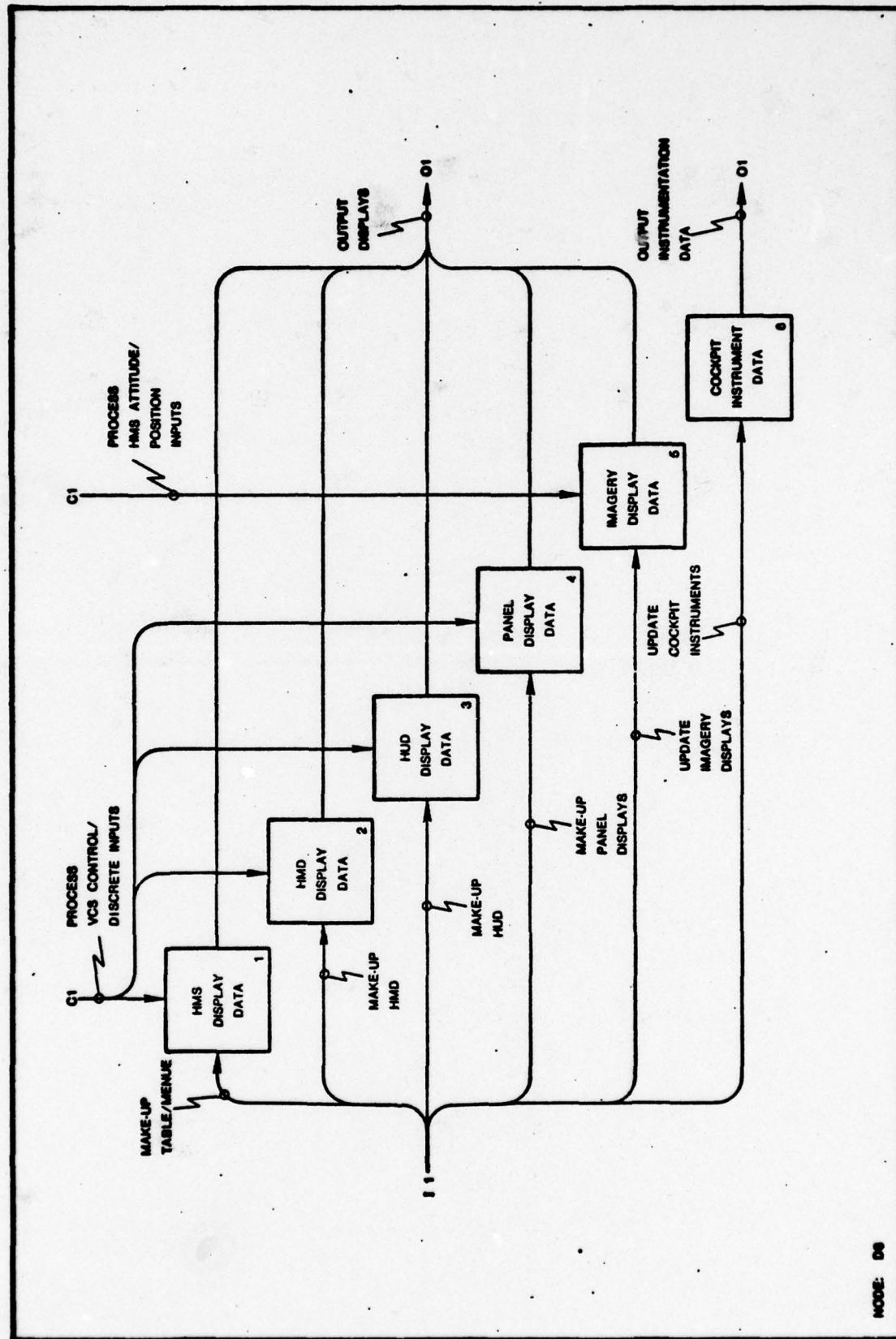


Figure 39. Simulation Displays

MODE: D8



D6 Text: The appropriate simulation display data (1, 2, 3, 4, 5) is formatted for display (11) and then output (01). The VCS control and discrete inputs (C1), such as a request for a specific display, control the display modes. The HMS attitude and position inputs (5C1) control the field-of-view of the imagery displays (501). The cockpit instrument data (6) is updated (611) and output (601) to the respective instruments. For example, the vehicle altitude is formatted and output to the altimeter.

### Summary

This chapter presented, for VCASS, the activity and data models, which are the formal specifications of the system. These specifications are the basis from which the software design in chapter IV is developed. Since none of the activity names, data names or arrow labels are binding, some of corresponding names and labels in the next chapter have been changed to aid in the understanding of the software design.

#### IV. Software Design

##### Introduction

The purpose of the design phase is to define the functions and operations necessary to satisfy the system requirements. The intent of this thesis is to design the software structure which will be used in implementing the VCASS system. As stated in Chapter I this thesis presents only a part of the design phase.

The technique utilized is called structured design. Structured design is a set of general program design considerations and techniques for making coding, debugging, and modification easier, faster, and less expensive by reducing complexity (Ref 8: 115). It simplifies the software by dividing it into modules which can be implemented and modified with minimal effects on other parts of the system. The criteria for evaluating the design is measured by the following attributes: (1) coupling, (2) cohesion, (3) span-of-control, and (4) scope-of-effect (Ref 10: 340).

Structured design starts by stating the problem as a data flow graph (or bubble chart). This bubble chart represents the conceptual ideas conveyed by the formal functional specifications (in Chapter III). A first cut is obtained by factoring the bubble chart into a tree structure known as a "structure chart". The resulting design is then evaluated, and revised, according to the above attributes. Interactions, between the structure-to-bubble chart and bubble-to-structure chart further refine the design. The resulting bubble chart and structured charts are presented in this chapter.



### Bubble Chart

A bubble chart transforms "conceptual inputs" into "conceptual outputs" and consists of "transforms" connected by labeled arrows. These "transforms", which are the basic elements of the bubble chart, are represented by circles labeled with short description of the transformations.

The asterisk ("\*") is used to indicate an "and" relationship between data elements. The "ring-sum" operator ("@") is used to denote "exclusive-or" relationships between data elements. Bubbles and arrows are drawn to represent input branches and output branches. These branches are connected by bubbles which are the "central transforms" of the system. The "central transforms" converts the "conceptual inputs" into "conceptual outputs". Identified on the bubble charts are the "afferent" and "efferent" data elements. An "afferent" data element is an input to a "central transform"; and an "efferent" data element is an output from a "central transform". The dashed lines and the dashed circle on this particular bubble chart represent a portion of the overall VCASS system software design which is not contained in this thesis, but is being modeled separately (Ref 9).

The final bubble chart for the VCASS system software design is shown in Figure 1 and represents the conceptual ideas conveyed by the formal functional specifications in Chapter III.

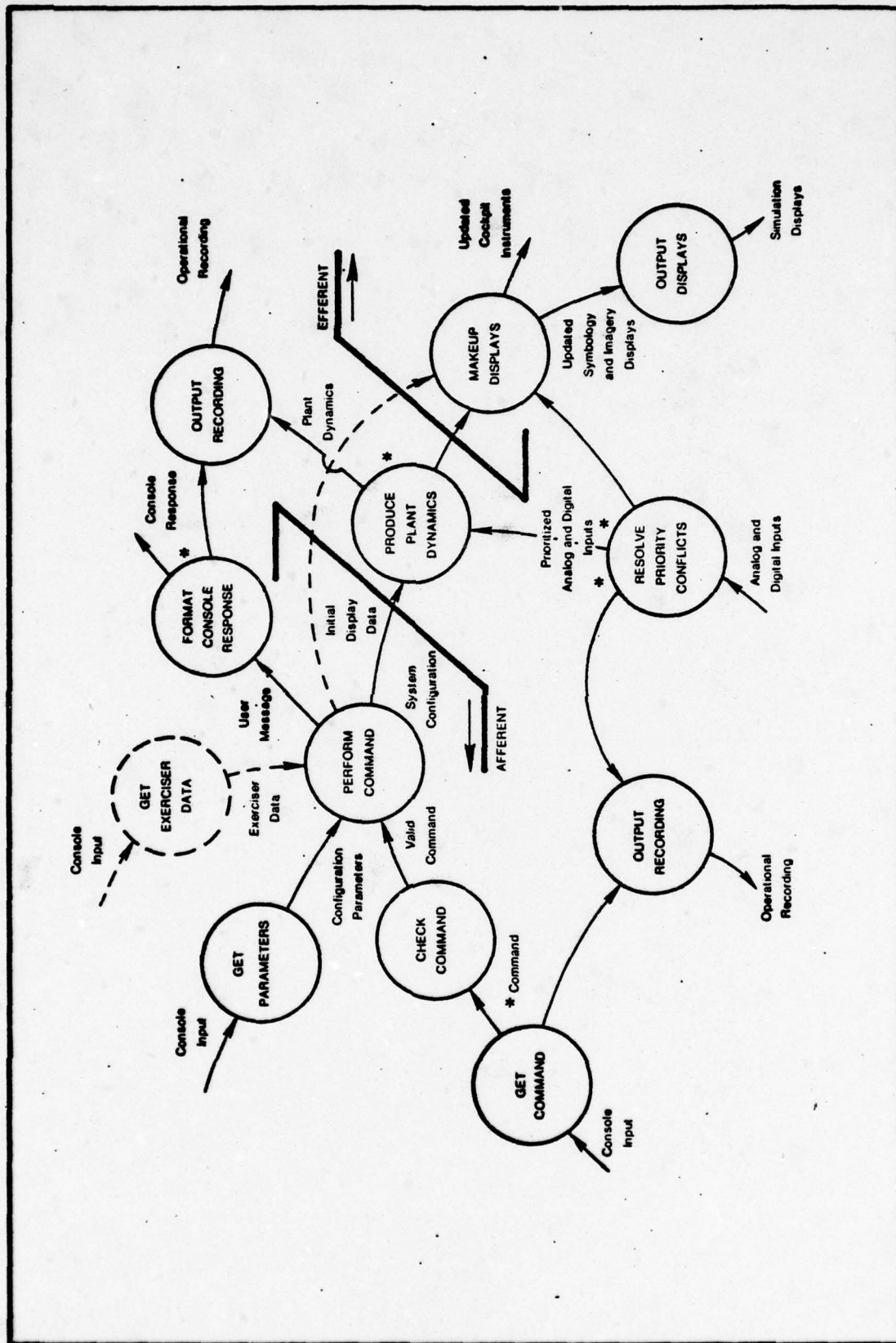


Figure 40. Final Bubble Chart

Bubble Chart Text: The transform 'Get Command' obtains commands from console input. These commands are checked for validity and recorded as part of the system history. The valid commands are either Operational or Exerciser commands. The Exerciser commands are processed by the Symbology Exerciser which is contained within the transform 'Perform Command'. The Symbology Exerciser allows for the building and modifying of the aircraft symbology display formats. The Exerciser command parameters are obtained by the transform 'Get Exerciser Data'. The aircraft symbology display formats (initial display data) are used to makeup the different aircraft displays. The Operational commands are processed by the transform 'Perform Command' and used to configure the system with respect to system-options, vehicle, weapons, cockpit, target, environment and display formats. When configuration parameters are required, 'Perform Command' generates a user message requesting the parameters. This request (user message) is formatted and output to the operators console. It is also recorded as a part of the system history.

Once the system is configured, the transform 'Produce Plant Dynamics' uses the system configuration parameters, analog inputs, and digital inputs to produce and update the plant dynamics (vehicle, target, and weapons state variables). When in a dual cockpit configuration, it is possible to receive the same type inputs from both cockpits. These multiple analog and digital input conflicts are resolved by the transform 'Resolve Priority Conflicts'. The plant dynamics, analog inputs, digital inputs, and the initial display data are used to makeup and update the displays and cockpit instruments. The plant dynamics along with the analog and digital inputs are also recorded as part of the system history.



### Structure Charts

The structure charts for the VCASS system software design are shown in Figures 41 through 61. They are organized as a sequence of charts with the associated documentation. The documentation includes the input/output tables (interface between modules) and a description of the modules. To aid the reader in understanding the design, a brief discussion of how to read structured charts can be found in Appendix B.

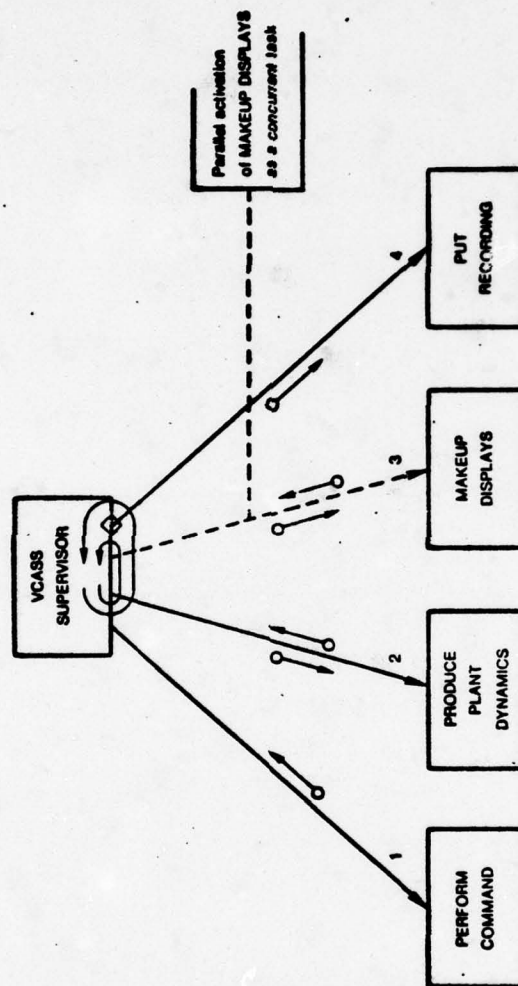


Figure 41. Top Level Structure

**Table I**  
**Parameters for Top Level Structure**

INPUT	OUTPUT
1 -----	System configuration data
2 HMS input data, System Configuration	Plant dynamics (vehicle, weapons, and target state variables; attack performance)
3 System configuration data, Plant dynamics (vehicle, weapons, and target state variables; attack performance)	HMS input data, Recording data (located in recording buffer)
4 Plant dynamics (vehicle, weapons, and target state variables; attack performance)	-----



### Module Descriptions for Top Level Structure

VCASS SUPERVISOR. This is the executive module. Its function is to control the overall operation of the simulator. When the executive is invoked, it will load and pass control to the top-level "afferent" module, 'PERFORM COMMAND', which processes user commands. When control is returned to the executive, the system enters the Operational mode. During this mode, the executive coordinates the concurrent tasks of producing the plant dynamics and updating and outputting the displays to the various hardware devices. The executive, also, periodically initiates the recording of predefined operational data.

PERFORM COMMAND. This module is designed to obtain a command from the operator console, determine the type, and perform the required operation by invoking the appropriate subordinate module. User commands are of two types, EXERCISER commands or OPERATIONAL commands. When an EXERCISER command is received, control is passed to the 'SYMBOLIC EXERCISER'. When an OPERATIONAL command is received, the system is configured according to the parameters entered, and control is then returned to the executive.

PRODUCE PLANT DYNAMICS. This module controls the production and updating of vehicle, weapons, and target dynamics; and controls the computation of the attack performance. The vehicle dynamics are continuously updated while the weapons and target dynamics are updated according to their respective mode selection. The attack performance is computed only when the vehicle is in the attack mode. This module, also, receives and processes plant discrete inputs: vehicle initialization, weapons mode selection, and target mode selection inputs.

MAKEUP DISPLAYS. This module controls the make-up and updating of all simulation displays and cockpit instrumentation. It also receives and processes the VCS control and discrete inputs. It is assumed that this module and all of its subordinates are to operate on a separate computer concurrently with 'PRODUCE PLANT DYNAMICS' and 'PUT RECORDING'. This module was placed within an iteration loop to indicate that data is systematically passed between it and 'VCASS SUPERVISOR'. The implementation of this transfer (whether by interrupt, direct memory access (DMA), etc.) has not been determined at this time.

PUT RECORDING. This module is periodically called by the executive to put the current plant dynamics into the recording buffer, and then dump that buffer to a recording device. All operator inputs and outputs, analog and digital inputs, plant state variables, and attack performance are contained in the recording buffer.

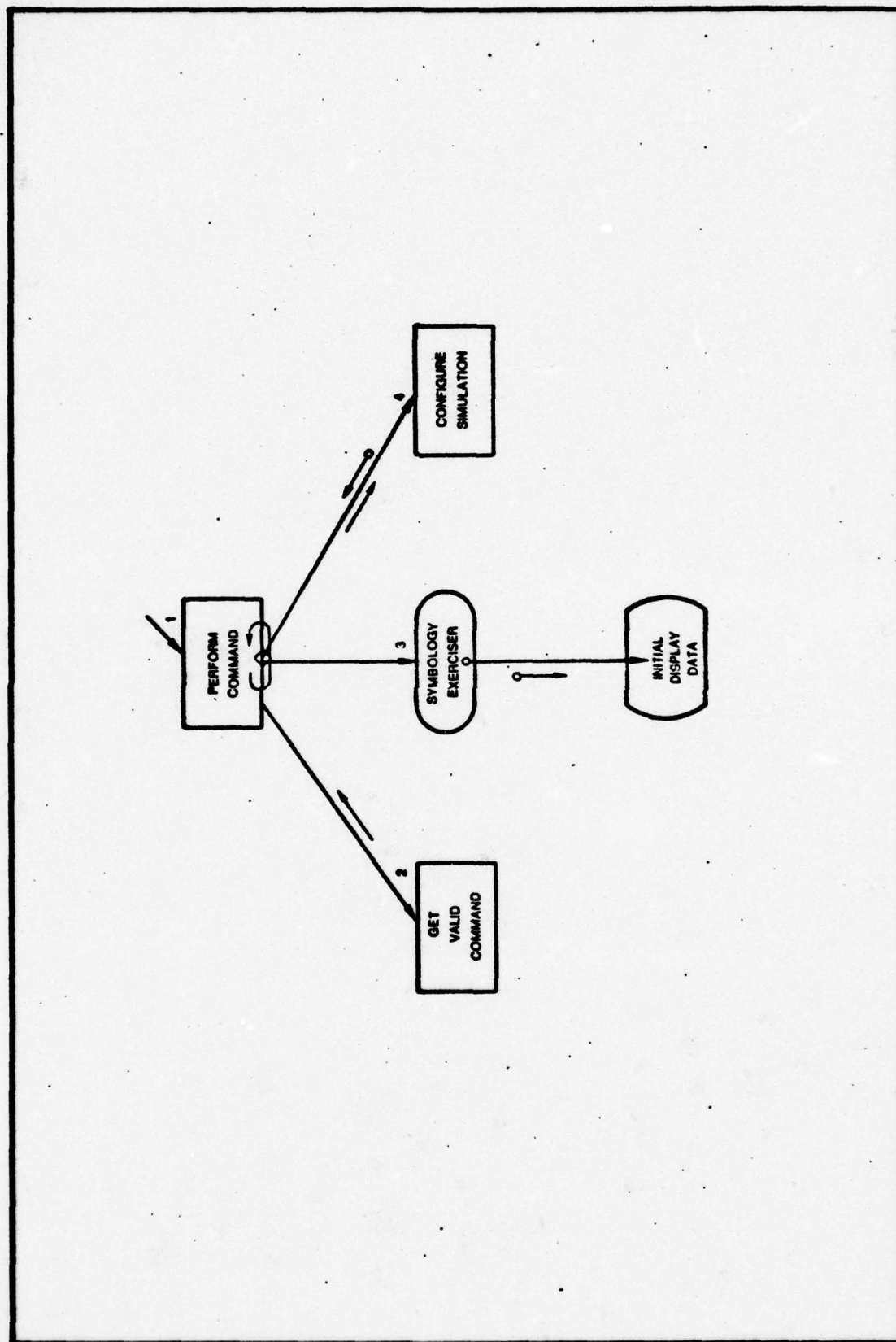


Figure 42. PERFORM COMMAND Branch



Table II  
Parameters for PERFORM COMMAND Branch

INPUT	OUTPUT
1 -----	System configuration data
2 -----	<u>EXERCISER or OPERATIONAL command</u> , Configuration parameters, Storage device
3 -----	-----
4 <u>Configuration parameters, Storage device</u>	System configuration data

### Module Descriptions for PERFORM COMMAND Branch

PERFORM COMMAND. This module is designed to obtain a command from the operators console, determine the type, and perform the required operation by invoking the appropriate subordinate module. User commands are of two types, EXERCISER commands or OPERATIONAL commands. When an EXERCISER command is received, control is passed to the 'SYMBOLGY EXERCISER'. When an OPERATIONAL command is received, the system is configured according to the parameters entered, and control is then returned to the executive.

GET VALID COMMAND. When the system is initiated, this module outputs a message to the user stating that the system is "up" and requesting the user to input a command. This module then obtains the command, records it, and error checks it. If the command is valid, it is passed to the superordinate module. If the command is invalid, an error message is output to the user, and this module then waits for another command to be entered.

SYMBOLGY EXERCISER. The 'SYMBOLGY EXERCISER' is invoked when a valid EXERCISER command is received. The 'SYMBOLGY EXERCISER' is a separate and independent subsystem of the VCASS simulator. It is used to dynamically build and modify the aircraft symbology display formats. The design of the 'SYMBOLGY EXERCISER' can be found in Ref 9.

CONFIGURE SIMULATION. When a valid OPERATIONAL command is received, this module is called to setup the system configuration. This module requires that the system be configured in a prescribed order. The

system-options are setup first with vehicle, weapons, cockpit, target, environment, and display configurations following in the order mentioned. Control is returned to the superordinate module when an END SYSTEM CONFIGURATION command is received.



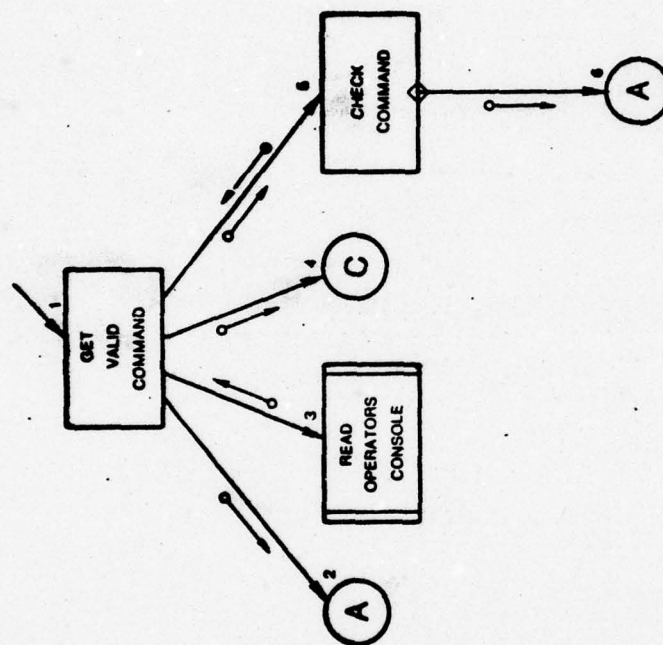


Figure 43. GET VALID COMMAND Branch

Table III  
Parameters for GET VALID COMMAND Branch

INPUT	OUTPUT
1. -----	<u>EXERCISER or OPERATIONAL command, Configuration parameters, Storage device</u>
2 Command request message	-----
3 -----	User command (with or without parameters)
4 User command, Recording buffer location	-----
5 User command	<u>Error Flag</u>
6 Command error message	-----

### Module Descriptions for GET VALID COMMAND Branch

GET VALID COMMAND. When the system is initiated, this module outputs a message to the user stating that the system is "up" and requesting the user to input a command. This module then obtains the command, records it, and error checks it. If the command is valid, it is passed to the superordinate module. If the command is invalid, an error message is output to the user, and this module then waits for another command to be entered.

READ OPERATORS CONSOLE. This is a system supplied module to input data from the operator console. It inputs, assembles, and returns a complete command to its superordinate.

CHECK COMMAND. This module checks a user command against a predefined list of commands for validity. If the command is valid, control is returned to the superordinate module. If the command is invalid, an error message is output to the user and an error flag is passed back to the superordinate module.



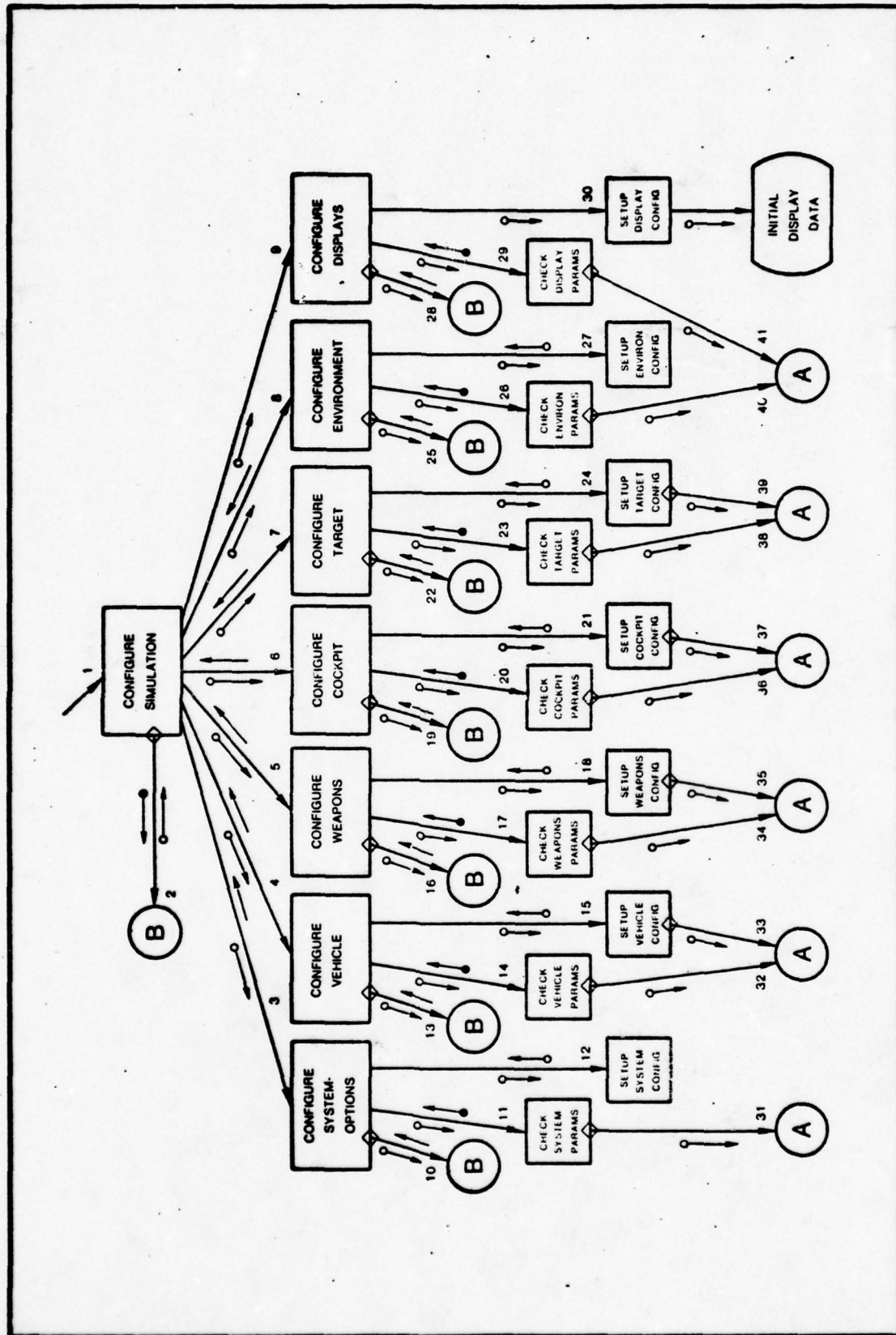


Figure 44. CONFIGURE SIMULATION Branch

Table IV  
Parameters for CONFIGURE SIMULATION Branch

INPUT	OUTPUT
1 Configuration parameters, Storage device	System configuration data
2 <u>Storage device</u>	Configuration parameters
3 System-Options configuration parameters	System-Options configuration, <u>END SYSTEM CONFIGURATION command</u>
4 Vehicle configuration parameters, System-Options configuration	Vehicle configuration, <u>END SYSTEM CONFIGURATION command</u>
5 Weapons configuration parameters, System-Options configuration, Vehicle configuration	Weapons configuration, <u>END SYSTEM CONFIGURATION command</u>
6 Cockpit configuration parameters, System-Options configuration, Vehicle configuration, Weapons configuration	Cockpit configuration, <u>END SYSTEM CONFIGURATION command</u>
7 Target configuration parameters, System-Options configuration	Target configuration, <u>END SYSTEM CONFIGURATION command</u>
8 Environment configuration parameters	Environment configuration, <u>END SYSTEM CONFIGURATION command</u>
9 Display configuration parameters	-----
10,13,16,19,22,25,28 Respective parameter request	Respective configuration parameters, Respective <u>END CONFIGURATION command or END SYSTEM CONFIGURATION command</u>

Table IV (Cont.)  
Parameters for CONFIGURE SIMULATION Branch

INPUT		OUTPUT
11,14,17,20,23,26,29	Respective configuration parameters	Error Flag
12	System-Options configuration parameters	System-Options configuration
15	Vehicle configuration parameters, System-Options configuration	Vehicle configuration
18	Weapons configuration parameters, System-Options configuration, Vehicle configuration	Weapons configuration
21	Cockpit configuration parameters, System-Options configuration, Vehicle configuration, Weapons configuration	Cockpit configuration
24	Target configuration parameters, System-Options configuration	Target configuration
27	Environment configuration parameters	Environment configuration
30	Display configuration parameters	---
31,32,34,35,38,40,41	Respective parameter error message	---
33,35,37,39	Respective set-up error message	---



### Module Descriptions for CONFIGURE SIMULATION Branch

CONFIGURE SIMULATION. When a valid OPERATIONAL command is received, this module is called to setup the system configuration. This module requires that the system be configured in a prescribed order. The system-options are setup first with vehicle, weapons, cockpit, target, environment, and display configurations following in the order mentioned. Control is returned to the superordinate module when an END SYSTEM CONFIGURATION command is received.

CONFIGURE SYSTEM-OPTIONS, CONFIGURE VEHICLE, CONFIGURE WEAPONS, CONFIGURE COCKPIT, CONFIGURE TARGET, CONFIGURE ENVIRONMENT, CONFIGURE DISPLAYS. Each of these modules control the configuration set-up of a specific part of the system. If the respective configuration parameters were not previously entered with the OPERATIONAL command or are not to be obtained from a specified storage device, these modules, in turn, request the user to enter them. Once entered they are checked for validity and then put into the proper format for use by the system. Some of the configuration affect the set-up of others, and, therefore, must be cross-checked. For example, the type of aircraft entered under vehicle configuration will affect what type of weapons which can be entered under weapons configuration.

CHECK SYSTEM PARAMETERS, CHECK VEHICLE PARAMETERS, CHECK WEAPONS PARAMETERS, CHECK COCKPIT PARAMETERS, CHECK TARGET PARAMETERS, CHECK ENVIRONMENT PARAMETERS, CHECK DISPLAY PARAMETERS. These modules check the validity of their respective configuration parameters entered. Such checks as allowable options, out-of-range values, and proper format are made. If an error is detected, a message, informing the user

of the error and requesting re-entry of the parameter, is output to the operator console.

SETUP SYSTEM CONFIGURATION, SETUP VEHICLE CONFIGURATION, SETUP WEAPONS CONFIGURATION, SETUP COCKPIT CONFIGURATION, SETUP TARGET CONFIGURATION, SETUP ENVIRONMENT CONFIGURATION, SETUP DISPLAY CONFIGURATION.

After the configuration parameters have been validity checked, these modules place them in the proper format for use by the system. The 'VEHICLE', 'WEAPONS', 'COCKPIT', and 'TARGET SETUP CONFIGURATION' modules must crosscheck their respective configuration parameters against previously set-up configurations. If a crosscheck error is found, a message is output to inform the user.

**NOTE:** The modules 'CONFIGURE SYSTEM-OPTIONS' and 'CONFIGURE DISPLAYS' were added to the functional requirements late in the design phase. Thus there is no mention of them in the requirements definition given in Chapters II and III.

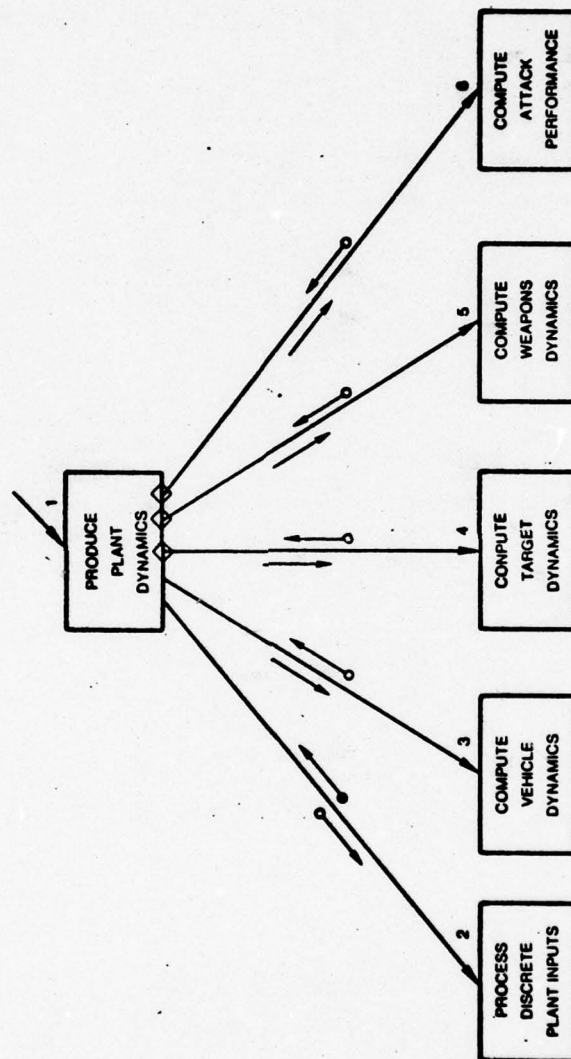


Figure 45. PRODUCE PLANT DYNAMICS Branch



**Table V**  
**Parameters for PRODUCE PLANT DYNAMICS Branch**

INPUT	OUTPUT
1 HMS input data, System configuration	Plant dynamics (vehicle, weapons, and target state variables; attack performance)
2 Cockpit configuration	<u>Discrete plant inputs (vehicle initialization inputs, target mode selection, weapons mode selection)</u>
3 <u>Vehicle initialization inputs</u> , HMS input data, <u>Vehicle configuration</u> , Environment configuration, Cockpit configuration	Vehicle state variables
4 <u>Target mode selection</u> , Target configuration, Vehicle state variables	Target state variables
5 <u>Weapons mode selection</u> , Weapons configuration, Cockpit configuration, Vehicle state variables, Target state variables	Weapons state variables
6 Weapons mode, Environment configuration, Target state variables, Weapons state variables	Attack performance

### Module Descriptions for PRODUCE PLANT DYNAMICS Branch

PRODUCE PLANT DYNAMICS. This module controls the production and updating of vehicle, weapons, and target dynamics; and controls the computation of the attack performance. The vehicle dynamics are continuously updated while the weapons and target dynamics are updated according to their respective mode selection. The attack performance is computed only when the vehicle is in the attack mode. This module, also, receives and processes plant discrete inputs: vehicle initialization, weapons mode selection, and target mode selection inputs.

PROCESS DISCRETE PLANT INPUTS. When this module is initiated, it calls a subordinate module to obtain the discrete plant inputs. Discrete plant inputs can occur from the following inputs: vehicle initialization, weapons mode selection, and target mode selection inputs. These inputs are error checked, prioritized, recorded, and passed to the superordinate module.

COMPUTE VEHICLE DYNAMICS. This module controls the initialization and updating of the vehicle state variables. The vehicle initialization module is called when initialization inputs are received. The air, takeoff/landing, and ground model modules are called to compute the vehicle state variables required to simulate flight, takeoff, landing, and ground operations respectively. The takeoff/landing model is called when the vehicle is in the takeoff or landing modes. The ground model is called when the vehicle altitude is zero or near zero.

COMPUTE TARGET DYNAMICS. Using the target mode selection input, this module determines the target mode of operation (canned mode, semi-smart mode, smart mode). The corresponding subordinate module is then activated to control that mode.

COMPUTE WEAPONS DYNAMICS. This module determines the weapons mode of operation (gun mode, AAM mode, AG mode) and activates the corresponding subordinate module to control that mode. When a change in weapons mode is made, this module initializes the weapons associated with the new mode.

COMPUTE ATTACK PERFORMANCE. Whenever the vehicle is in the attack mode, this module is called to determine the accuracy of either air-to-air weapons deliveries or air-to-ground weapons deliveries. This module determines the delivery mode and calls the corresponding subordinate module to perform the evaluation.



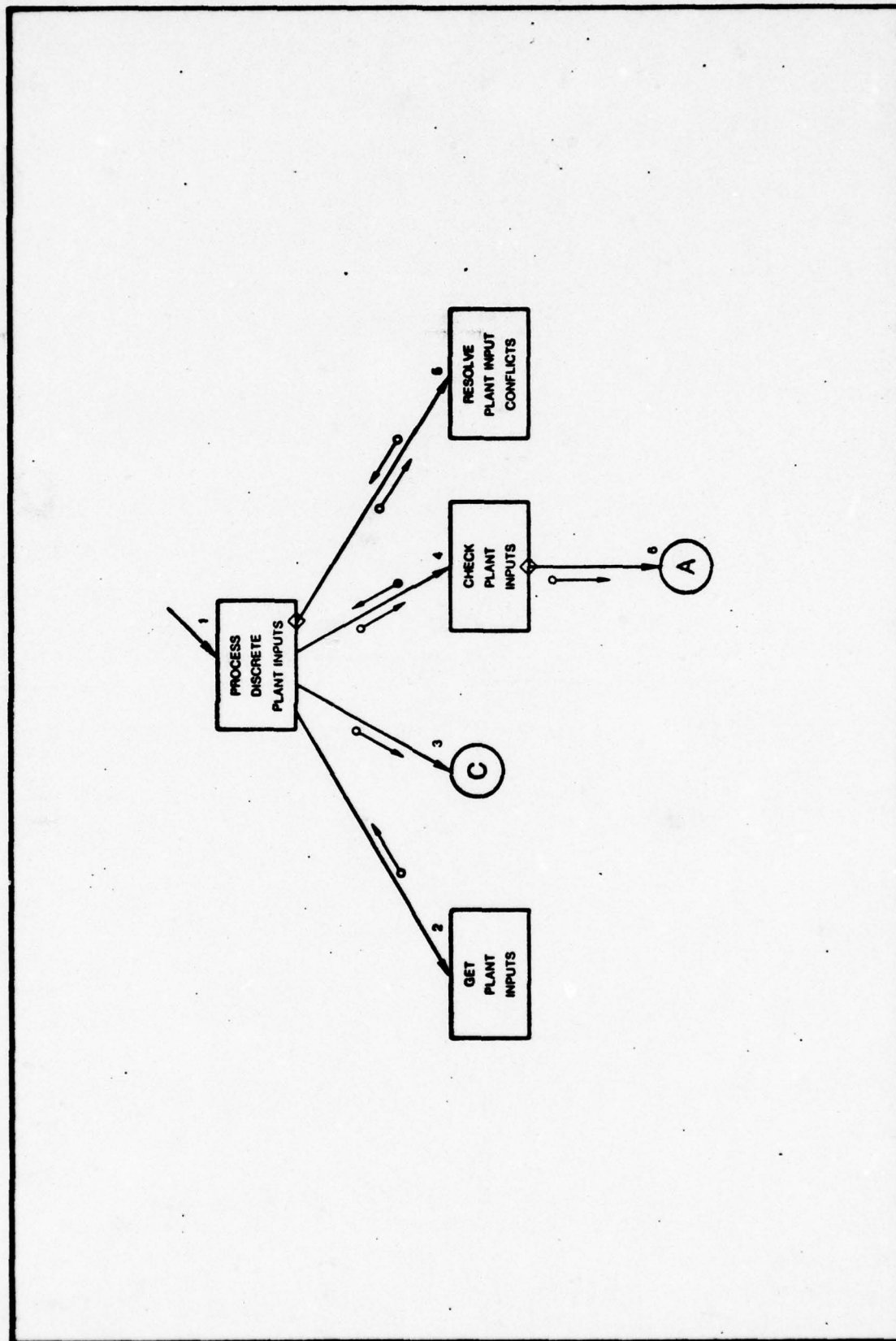


Figure 46. PROCESS DISCRETE PLANT INPUTS Branch

Table VI  
Parameters for PROCESS DISCRETE PLANT INPUTS Branch

INPUT	OUTPUT
1 Cockpit configuration	<u>Discrete plant inputs (vehicle initialization inputs, target mode selection, weapons mode selection)</u>
2 -----	Discrete plant inputs
3 Discrete plant inputs, recording buffer location	-----
4 Discrete plant inputs	<u>Error flag</u>
5 Multiple discrete plant inputs, Cockpit configuration	Prioritized discrete plant inputs
6 Discrete input error message	-----

### Module Descriptions for PROCESS DISCRETE PLANT INPUTS Branch

PROCESS DISCRETE PLANT INPUTS. When this module is initiated, it calls a subordinate module to obtain the discrete plant inputs. Discrete plant inputs can occur from the following inputs: vehicle initialization, weapons mode selection, and target mode selection inputs. These inputs are error checked, prioritized, recorded, and passed to the superordinate module.

GET PLANT INPUTS. This module is called to poll the discrete plant input lines to determine if any inputs have occurred. It then gets those inputs from the corresponding input buffer and passes them to the superordinate module.

CHECK PLANT INPUTS. The discrete plant inputs are error checked by this module. If an error is found, the inputs are discarded and a message, informing the operator of the error, is output to the operator console.

RESOLVE PLANT INPUT CONFLICTS. With a dual cockpit configuration, it is possible to receive the same type inputs from both cockpits. This module determines which inputs have priority and discards the others.





Table VII

## Parameters for COMPUTE VEHICLE DYNAMICS Branch

INPUT	OUTPUT
1 Vehicle initialization inputs, HMS input data, Vehicle configuration, Environment configuration, Cockpit configuration	Vehicle state variables
2 Vehicle initialization inputs	Vehicle state variables
3,4,5 Vehicle configuration, Environment configuration, Cockpit configuration	Vehicle state variables
6,7,8 Vehicle configuration, Cockpit configuration	Vehicle parameters
9 Cockpit configuration	Stick, Rudder, and Throttle inputs
10 Throttle inputs, Vehicle configuration	Engine model parameters
11 Stick and Rudder inputs, Vehicle configuration	Stick/Rudder model parameters
12 -----	Stick, Rudder, and Throttle inputs
13 Stick, Rudder, and Throttle inputs, Recording buffer location	-----
14 Stick, Rudder, and Throttle inputs	<u>Error Flag</u>
15 Multiple Stick, Rudder, and Throttle inputs, Cockpit configuration	Prioritized Stick, Rudder, and Throttle inputs
16 Vehicle input error message	-----

### Module Descriptions for COMPUTE VEHICLE DYNAMICS Branch

COMPUTE VEHICLE DYNAMICS. This module controls the initialization and updating of the vehicle state variables. The vehicle initialization module is called when initialization inputs are received. The air, takeoff/landing, and ground model modules are called to compute the vehicle state variables required to simulate flight, takeoff, landing, and ground operations respectively. The takeoff/landing model is called when the vehicle is in the takeoff or landing modes. The ground model is called when the vehicle altitude is zero or near zero.

INITIALIZE VEHICLE. Whenever vehicle initialization inputs are received, this module is called to initialize the vehicle position, angles, and rates.

COMPUTE AIR MODEL. This module provides the mathematical equations, variable interrelations, and time integrations required to model the configured vehicle during flight operation. It is called when the vehicle is in the enroute mode and when it is not on the ground.

COMPUTE TAKEOFF/LANDING MODEL. This module is an expanded version of the air model, and provides for vehicle response to takeoff and landing conditions. It is called when the vehicle is in the takeoff or landing modes and when it is not on the ground.

COMPUTE GROUND MODEL. This module is an expanded version of the air model, and provides for vehicle



response to taxi and crash conditions. The purpose of the ground model is to allow the pilot to "sense" that the vehicle is on the ground. It is called when the vehicle altitude is zero or near zero.

GET VEHICLE PARAMETERS. The computation of engine, stick, and rudder related parameters is controlled by this module. It obtains stick, rudder, and throttle inputs and passes them to the appropriate subordinate module for processing. The resultant parameters are passed to the calling superordinate module.

PROCESS VEHICLE INPUTS. This module is called to get the vehicle stick, rudder, and throttle inputs. These inputs are error checked, recorded, prioritized, and passed to the superordinate module.

COMPUTE ENGINE MODEL. The purpose of this module is to compute all engine related parameters for use by the appropriate aircraft model. Some pertinent parameters computed are thrust, thrust available, engine RPM, and engine pressure ratio.

COMPUTE STICK/RUDDER MODEL. This module translates stick and rudder displacements into control surface deflections for use by the appropriate aircraft model. Some pertinent parameters computed are elevator, aileron, and rudder deflections and trims.

GET VEHICLE INPUTS. This module is called to get flight control inputs from the proper input buffer. The inputs are passed to the superordinate module for processing.

CHECK VEHICLE INPUTS. The flight control inputs are error checked by this module. If an error is found, the inputs are discarded and a message, informing the operator of the error, is output to the operator console.

RESOLVE VEHICLE INPUT CONFLICTS. With a dual cockpit configuration, it is possible to receive the same type inputs from both cockpits. This module determines which inputs have priority and discards the others.

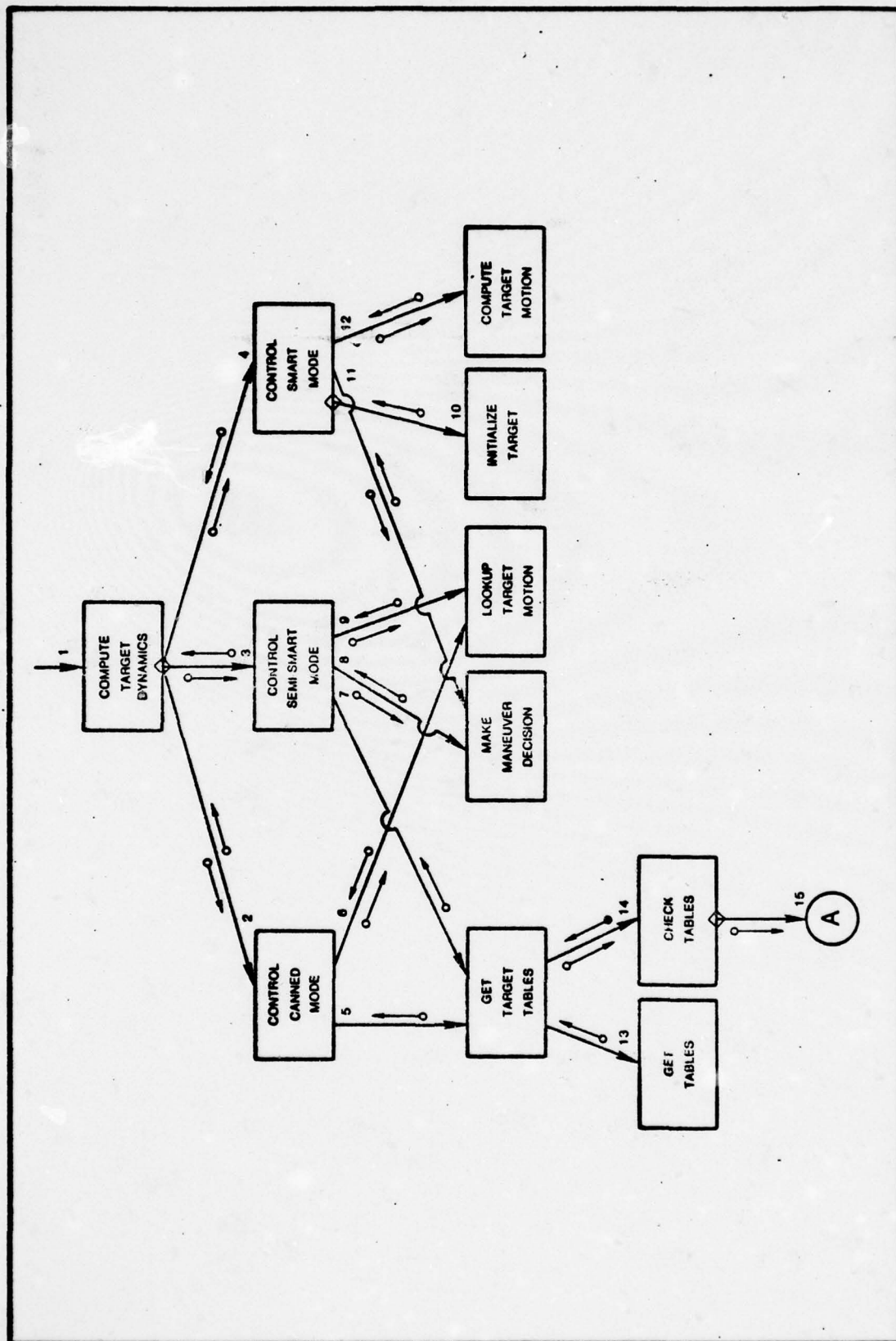


Figure 48. COMPUTE TARGET DYNAMICS Branch



Table VIII

## Parameters for COMPUTE TARGET DYNAMICS Branch

INPUT	OUTPUT
1 <u>Target mode selection</u> , Target configuration, Vehicle state variables	Target state variables
2 Target configuration	Target state variables
3,4 Target configuration, Vehicle state variables	Target state variables
5 -----	Target preprogrammed tables
6 Maneuver pointer, Target configuration, Target preprogrammed tables	Target state variables
7 -----	Target preprogrammed tables
8 Vehicle state variables, Target state variables	Maneuver pointer
9 Maneuver pointer, Target configuration, Target preprogrammed tables	Target state variables
10 -----	Target state variables
11 Vehicle state variables, Target state variables	Maneuver pointer
12 Maneuver pointer, Target configuration, Vehicle state variables	Target state variables
13 -----	Target preprogrammed tables
14 Target preprogrammed tables	<u>Error flag</u>
15 Target table input error message	-----

### Module Descriptions for COMPUTE TARGET DYNAMICS Branch

COMPUTE TARGET DYNAMICS. Using the target mode selection input, this module determines the target mode of operation (canned mode, semi-smart mode, smart mode). The corresponding subordinate module is then activated to control that mode.

CONTROL CANNED MODE. When first activated, this module calls a subordinate module to obtain preprogrammed target tables from an external storage device. Once the tables have been input, the target flight path is extracted from them and returned to the superordinate module as a set of target state variables.

CONTROL SEMI-SMART MODE. When first activated, this module calls a subordinate module to obtain preprogrammed target tables from an external storage device. Once the tables have been input, a maneuver pointer, based upon the state of the attacking vehicle, is obtained. This maneuver pointer is then used to control the extraction of the target flight path from the preprogrammed target tables.

CONTROL SMART MODE. This module controls the execution of a predefined algorithm to compute the target flight path. After the target is initialized, this module obtains a vehicle maneuver pointer, as is done when in the semi-smart mode. This maneuver pointer is then passed to a subordinate module which contains the algorithm to compute the target motion. A set of new target state variables is returned.

GET TARGET TABLES. This module controls the input of the target preprogrammed tables and checks them

for possible errors.

MAKE MANEUVER DECISION. Using the present state of the vehicle and target, this module calculates a maneuver pointer. This maneuver pointer allows the target motion to be based upon the attacking vehicle maneuvers, giving a "smart" capability.

LOOKUP TARGET MOTION. Based on the maneuver pointer, this module performs a table look-up in the target preprogrammed tables to obtain the new target state variables.

INITIALIZE TARGET. This module is called to initialize the target position, angles, and rates, establishing a base point from which to compute the target motion.

COMPUTE TARGET MOTION. This module contains a predefined algorithm to compute the target motion, based on the state of the attacking vehicle. A set of new target state variables is returned to the superordinate module.

GET TABLES. This module inputs the target preprogrammed tables from an external storage device. The tables are then passed to the superordinate module.

CHECK TABLES. The target preprogrammed tables are error checked by this module. If an error is found, the tables are discarded and a message, informing the operator of the error, is output to the operator console.



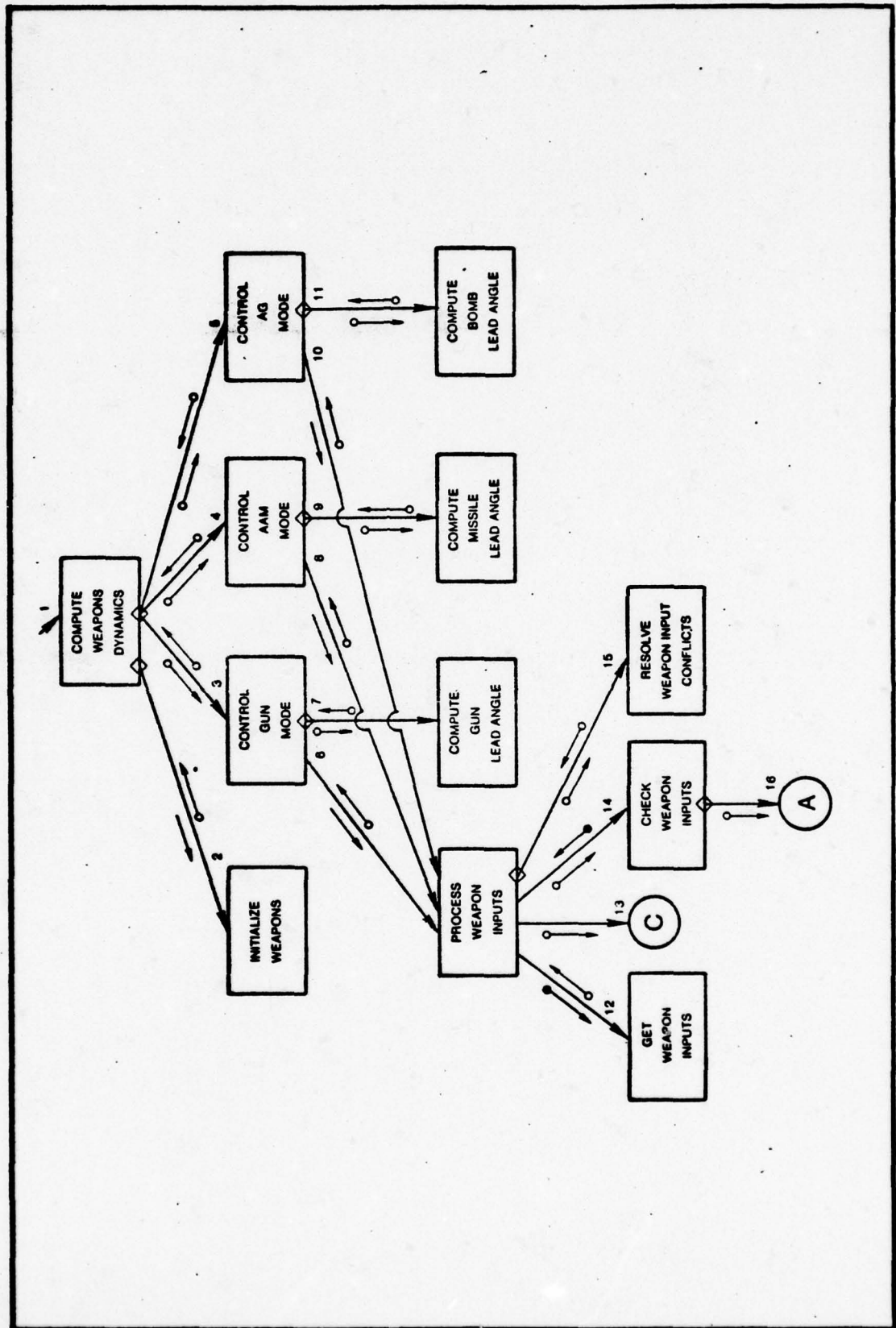


Figure 49. COMPUTE WEAPONS DYNAMICS Branch

Table IX

## Parameters for COMPUTE WEAPONS DYNAMICS Branch

INPUT	OUTPUT
1 <u>Weapons mode selection</u> , <u>Weapons configuration</u> , <u>Cockpit configuration</u> , <u>Vehicle state variables</u> , <u>Target state variables</u>	Weapons state variables
2 <u>Weapons mode</u> , <u>Weapons configuration</u>	Weapons initialization variables
3,4,5 <u>Weapons initialization variables</u> , <u>Weapons</u> <u>configuration</u> , <u>Cockpit configuration</u> , <u>Vehicle state variables</u> , <u>Target state</u> <u>variables</u>	Weapons state variables
6,8,10 <u>Weapons mode</u> , <u>Cockpit configuration</u>	Weapon inputs (trigger, release, etc.)
7,9,11 <u>Weapons configuration</u> , <u>Vehicle state</u> <u>variables</u> , <u>Target state variables</u>	Respective lead angle
12 <u>Input buffer location</u>	Weapon inputs
13 <u>Weapon inputs</u> , <u>Recording buffer location</u>	-----
14 <u>Weapon inputs</u>	<u>Error flag</u>
15 <u>Multiple weapon inputs</u> , <u>Cockpit configuration</u>	Prioritized weapon inputs
16 <u>Weapon input error message</u>	-----

Module Descriptions for COMPUTE WEAPONS DYNAMICS Branch

COMPUTE WEAPONS DYNAMICS. This module determines the weapons mode of operation (gun mode, AAM mode, AG mode) and activates the corresponding subordinate module to control that mode. When a change in weapons mode is made, this module initializes the weapons associated with the new mode.

INITIALIZE WEAPONS. Whenever a weapons mode is activated or a change in weapons mode is made, this module is called to initialize the weapons associated with the new mode. The initialization includes checking the weapons, setting triggers, and activating the sights.

CONTROL GUN MODE. This module controls the aircraft weapons gun mode. It obtains corresponding mode inputs, calculates gun lead angle if gun sight is activated, computes bullet trajectories when triggered, and updates associated weapons inventory.

CONTROL AAM MODE. This module controls the aircraft weapons air-to-air missile mode. It obtains corresponding mode inputs, calculates missile lead angle if missile sight is activated, computes missile trajectory when released, and updates the associated weapons inventory.

CONTROL AG MODE. This module controls the aircraft weapons air-to-ground mode. It obtains corresponding mode inputs, calculates bomb lead angle if bomb sight is activated, computes bomb trajectory when released, and updates the associated weapons inventory.



PROCESS WEAPON INPUTS. This module is called to get the weapon inputs that correspond with the current weapons mode. These inputs are error checked, recorded, prioritized, and passed to the superordinate module.

COMPUTE GUN LEAD ANGLE, COMPUTE MISSILE LEAD ANGLE, COMPUTE BOMB LEAD ANGLE. Contained in these modules are the mathematical equations necessary to calculate the corresponding weapon lead angles for the corresponding aiming sights.

GET WEAPON INPUTS. This module is called to get the weapon inputs from the input buffer specified. These inputs are passed to the superordinate module for processing.

CHECK WEAPON INPUTS. The weapon inputs are error checked by this module. If an error is found, the inputs are discarded and a message, informing the operator of the error, is output to the operator console.

RESOLVE WEAPON INPUT CONFLICTS. With a dual cockpit configuration, it is possible to receive the same type inputs from both cockpits. This module determines which inputs have priority and discards the others.

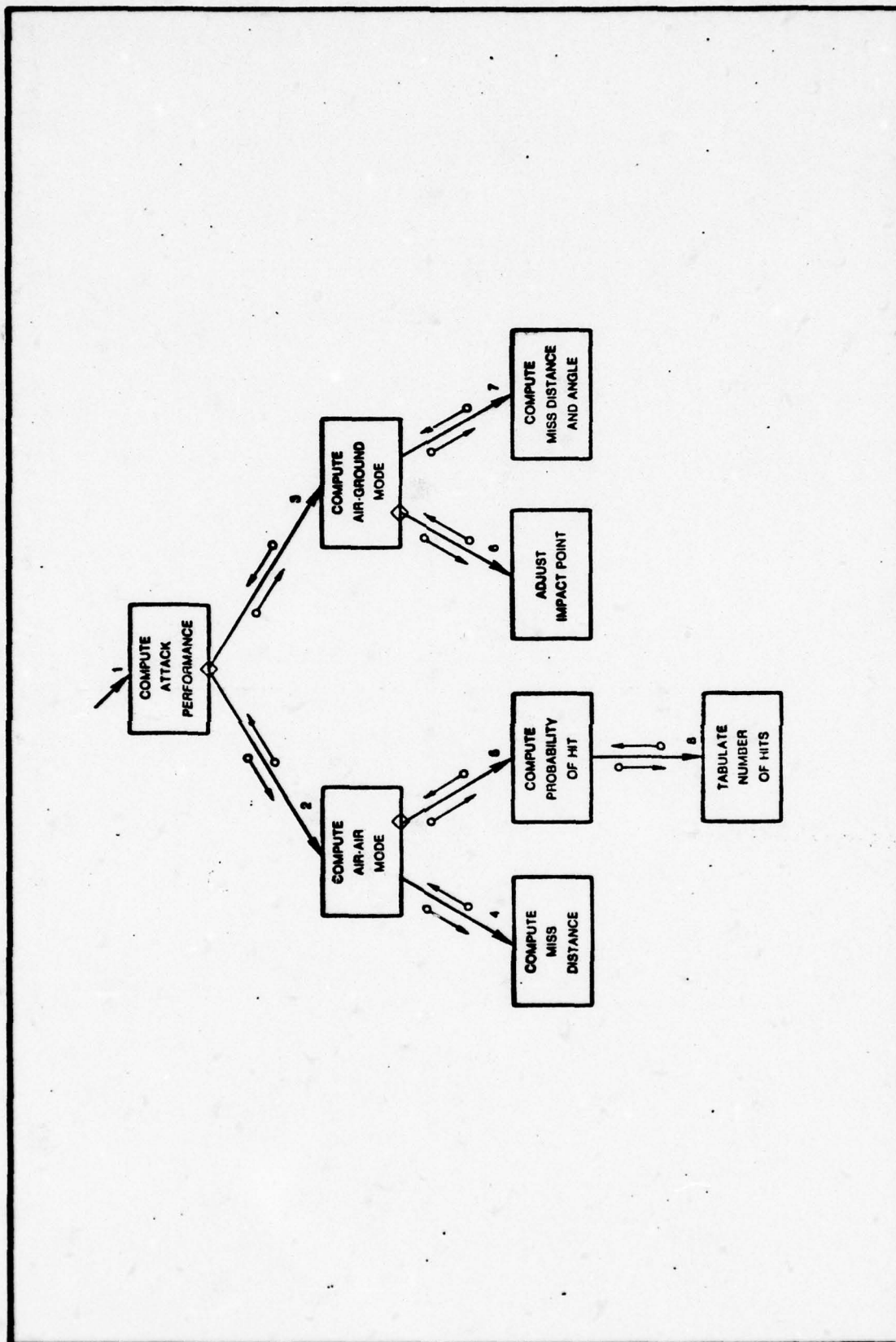


Figure 50. COMPUTE ATTACK PERFORMANCE Branch

Table X

Parameters for COMPUTE ATTACK PERFORMANCE Branch

INPUT	OUTPUT
1 Weapons mode, Environment configuration, Target state variables, Weapons state variables	Attack performance
2 Target state variables, Weapons state variables	Attack performance
3 Environment configuration, Target state variables, Weapons state variables	Attack performance
4 Target state variables, Weapons state variables	Minimum miss distance, Scoring frame
5 Scoring frame, Weapons state variables	Number of hits
6 Environment configuration, Weapons state variables	Impact point
7 Impact point, Target state variables, Weapons state variables	Miss distance and angle
8 Hit probability, Weapons state variables	Number of hits



### Module Descriptions for COMPUTE ATTACK PERFORMANCE Branch

COMPUTE ATTACK PERFORMANCE. Whenever the vehicle is in the attack mode, this module is called to determine the accuracy of either air-to-air weapons deliveries or air-to-ground weapons deliveries. This module determines the delivery mode and calls the corresponding subordinate module to perform the evaluation.

COMPUTE AIR-AIR MODE. Using the type of weapon released, trajectory of weapon released, and location of target, this module controls the evaluation and scoring of all air-to-air weapons.

COMPUTE AIR-GROUND MODE. Using the trajectory of the bomb released, location of the target, and wind effects, this module controls the evaluation and scoring of air-to-ground weapons.

COMPUTE MISS DISTANCE. This module calculates a weapon miss vector. If the weapon being used is the gun, the miss vector is transformed by a bullet dispersion coordinate system into a scoring frame.

COMPUTE PROBABILITY OF HIT. When the gun is being used, this module is called to calculate a single-shot hit probability and tabulate the number of possible and actual hits.

ADJUST IMPACT POINT. Whenever the wind speed is greater than zero, this module is called to adjust the impact point according to the effects of the moving air mass on the weapon.

COMPUTE MISS DISTANCE AND ANGLE. This module compares the predicted ground impact point to the known target location to compute a miss distance and angle.

TABULATE NUMBER OF HITS. Using the hit probability and the number of bullets fired, this module tabulates the number of possible and actual hits.

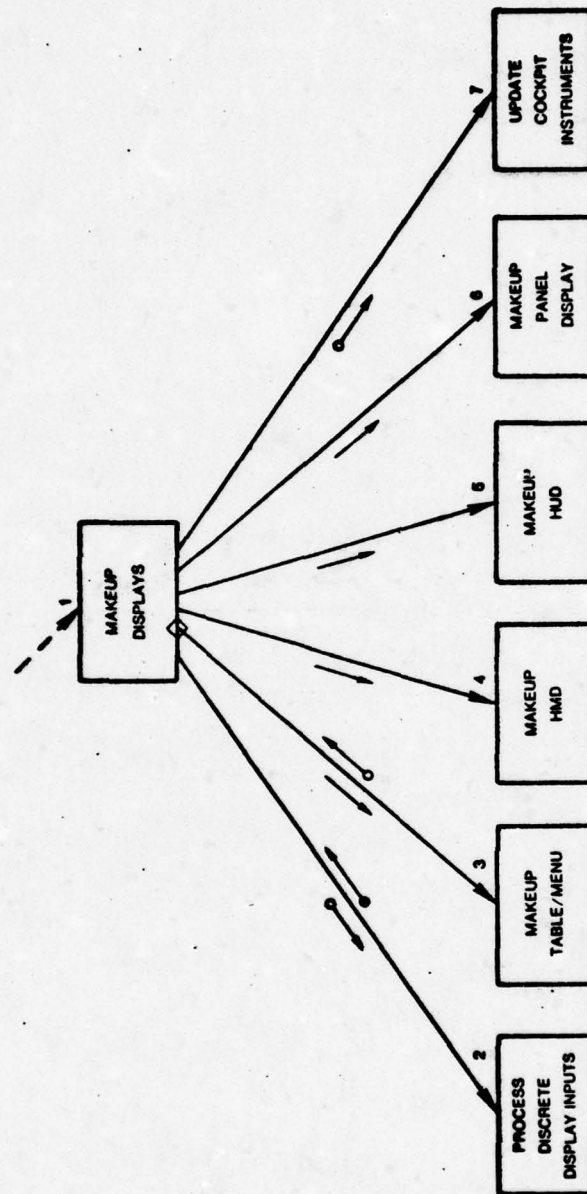


Figure 51. MAKEUP DISPLAYS Branch



Table XI

## Parameters for MAKEUP DISPLAYS Branch

INPUT	OUTPUT
1 System configuration data, Plant dynamics (vehicle, weapons, and target state variables; attack performance)	HMS input data, Recording data (located in recording buffer)
2 Cockpit configuration	<u>Discrete display inputs (VCS control and discrete inputs)</u>
3 <u>Table/Menu request</u> , Menu inputs, Cockpit configuration	HMS input data
4 <u>HMD display mode</u> , <u>Vehicle flight mode</u> , Plant dynamics (vehicle, weapons, and target state variables; attack performance), Environment configuration, Cockpit configuration	---
5 <u>Vehicle flight mode</u> , Vehicle state variables, Weapons state variables, Attack performance, Cockpit configuration	---
6 <u>Panel display mode</u> , <u>Vehicle flight mode</u> , Plant dynamics (vehicle, weapons, and target state variables; attack performance), Environment configuration, Cockpit configuration	---
7 Vehicle configuration, Cockpit configuration, Vehicle state variables	---

### Module Descriptions for MAKEUP DISPLAYS Branch

MAKEUP DISPLAYS. This module operates concurrently with other tasks in the system. It controls the make-up and updating of all simulation displays and cockpit instrumentation. It also receives and processes the VCS control and discrete inputs.

PROCESS DISCRETE DISPLAY INPUTS. When this module is activated, it calls a subordinate module to obtain the discrete display inputs. Discrete display inputs can occur from the following inputs: VCS control, discrete, and HMS inputs. These inputs are error checked, recorded, prioritized, and passed to the superordinate module.

MAKEUP TABLE/MENU. This module is called to process table/menu requests and menu inputs. It calls subordinate modules to determine if the request or inputs are valid and entered in the proper sequence. It dispatches the valid inputs to the 'UPDATE MENU' module and the valid requests to the 'ASSEMBLE TABLE/MENU' module. After the displays have been built or updated, this module calls a subordinate module to output these displays to the HMD device. The HMS graphics data is initialized when this module is first activated.

MAKEUP HMD. Since the HMD is oriented with respect to the operator line-of-sight, this module obtains HMS attitude and position inputs. This module then determines the HMD display mode, activates the corresponding display or displays, and passes the HMS inputs to the appropriate subordinate modules.

It is possible and likely that several of the HMD displays will be active at the same time, with the following exceptions: HUD and FSD, VVHD and TERRAIN PORTRAYAL, VVHD and BACKGROUND/ENVIRONMENT. After the display has been built or updated, this module calls a subordinate module to output the display to the HMD device. The HMD graphics data is initialized when this module is first activated.

MAKEUP HUD. When called, this module activates the HUD display that corresponds to the vehicle flight mode. After the display has been built or updated, this module calls a subordinate module to output the display to the HUD device. The HUD graphics data is initialized when this module is first activated.

MAKEUP PANEL DISPLAY. This module determines the panel display mode and activates the corresponding display or displays. It is possible and likely that several of the panel displays will be active at the same time, with the following exceptions: HUD and FSD, VVHD and TERRAIN PORTRAYAL, VVHD and BACKGROUND/ENVIRONMENT. After the display has been built or updated, this module calls a subordinate module to output the display to the panel. The panel graphics data is initialized when this module is first activated.

UPDATE COCKPIT INSTRUMENTS. This module is called to update the cockpit instruments from the respective vehicle state variables. This module calls subordinate modules to format and output the new instrument readings. The instruments to be updated are the clock, airspeed indicator, turn and slip indicator, elapsed time meter, magnetic compass, artificial horizon, directional gyro, vertical speed indicator,



altimeter, power indicator, fuel gauge, and gear/flap position indicator.

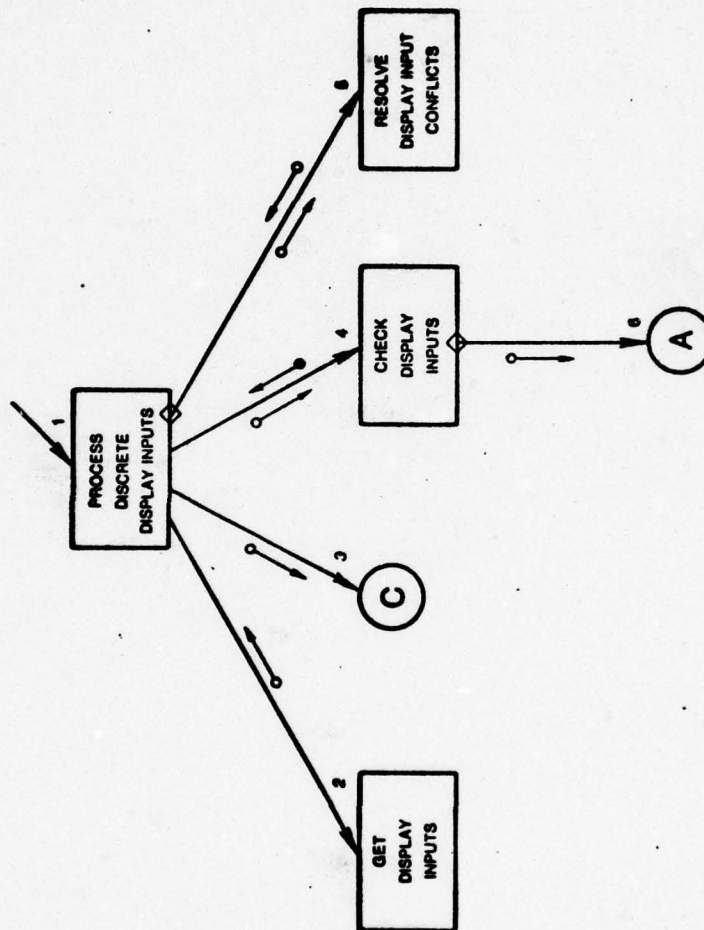


Figure 52. PROCESS DISCRETE DISPLAY INPUTS Branch

Table XII  
Parameters for PROCESS DISCRETE DISPLAY INPUTS Branch

INPUT	OUTPUT
1 Cockpit configuration	<u>Discrete display inputs (VCS control and discrete inputs)</u>
2 -----	Discrete display inputs
3 Discrete display inputs, Recording buffer location	-----
4 Discrete display inputs	<u>Error flag</u>
5 Multiple discrete display inputs, Cockpit configuration	Prioritized discrete display inputs
6 Discrete input error message	-----



### Module Descriptions for PROCESS DISCRETE DISPLAY INPUTS Branch

PROCESS DISCRETE DISPLAY INPUTS. When this module is activated, it calls a subordinate module to obtain the discrete display inputs. Discrete display inputs can occur from the following inputs: VCS control, discrete, and HMS inputs. These inputs are error checked, recorded, prioritized, and passed to the superordinate module.

GET DISPLAY INPUTS. This module is called to poll the discrete display input lines to determine if any have occurred. It then gets those inputs from the corresponding input buffer and passes them to the superordinate module.

CHECK DISPLAY INPUTS. The discrete display inputs are error checked by this module. If an error is found, the inputs are discarded and a message, informing the operator of the error, is output to the user terminal.

RESOLVE DISPLAY INPUT CONFLICTS. With a dual cockpit configuration, it is possible to receive the same inputs from both cockpits. This module determines which inputs have priority and discards the others.

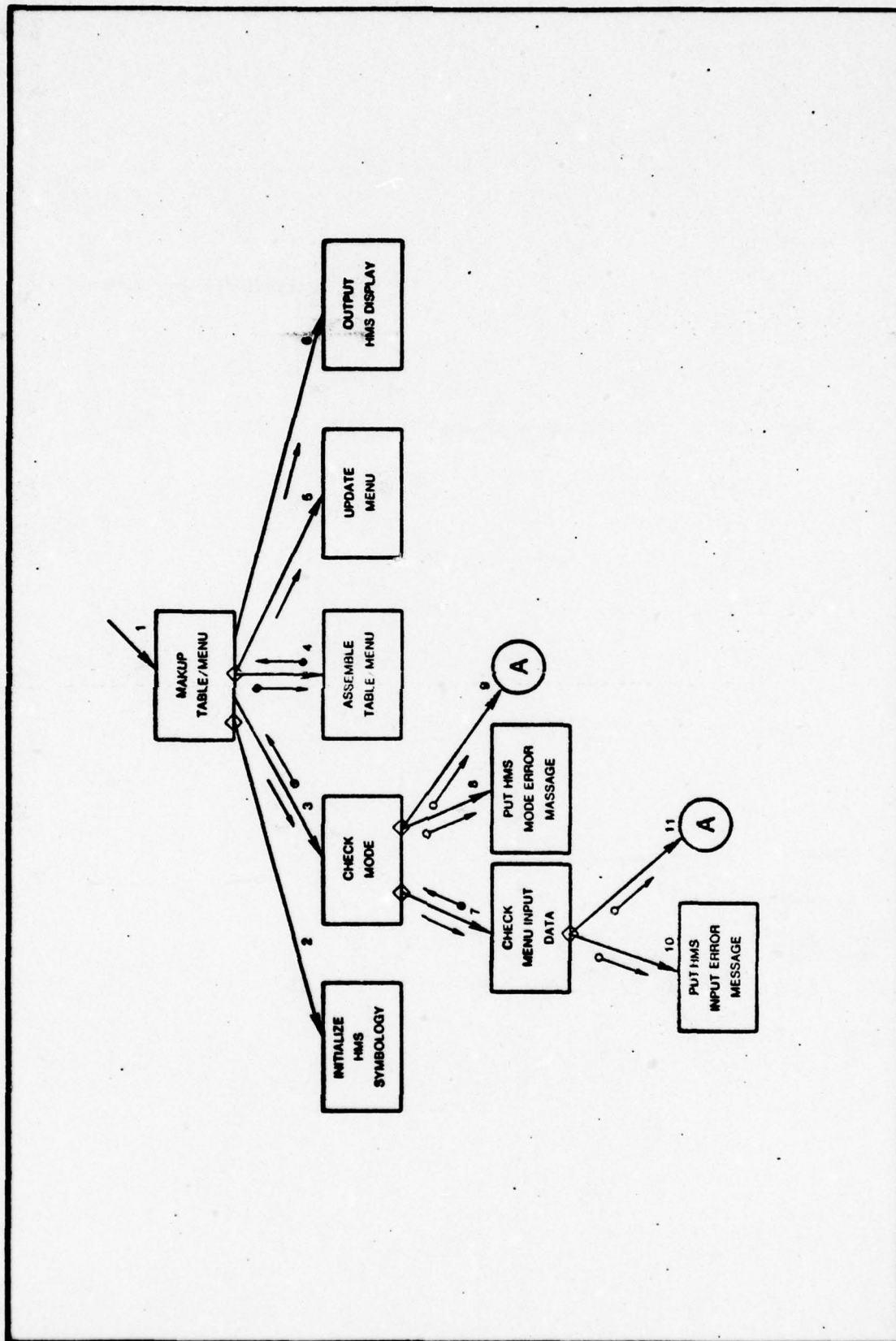


Figure 53. MAKEUP TABLE/MENU BRANCH

Table XIII  
Parameters for MAKEUP TABLE/MENU Branch

INPUT		OUTPUT
1	Table/Menu request, Menu inputs, Cockpit configuration	HMS input data
2	----	-----
3	Table/Menu request, Menu inputs, <u>Display active indicator</u>	<u>Input mode</u>
4	<u>Table/Menu request</u>	<u>Display active indicator</u>
5	Menu inputs, <u>Display active indicator</u>	-----
6	Cockpit configuration, <u>Display active indicator</u>	-----
7	Menu inputs, <u>Display active indicator</u>	<u>Error flag</u>
8	Input mode error message	-----
9	Input mode error message	-----
10	Menu input error message	-----
11	Menu input error message	-----



#### Module Descriptions for MAKEUP TABLE/MENU Branch

MAKEUP TABLE/MENU. This module is called to process table/menu requests and menu inputs. It calls subordinate modules to determine if the request or inputs are valid and entered in the proper sequence. It dispatches the valid inputs to the 'UPDATE MENU' module and the valid requests to the 'ASSEMBLE TABLE/MENU' module. After the displays have been built or updated, this module calls a subordinate module to output these displays to the HMD device. The HMS graphics data is initialized when this module is first activated.

INITIALIZE HMS SYMBOLOGY. This module is called to initialize the HMS graphics data. The purpose of this initialization is to setup the HMS displays in a form that can be readily updated and output to the display device.

CHECK MODE. This module is called to determine if the table/menu request or menu inputs are valid and have been entered in the proper sequence. If an error is found a message, informing the operator of the error, is output to the HMD and to the operator console.

ASSEMBLE TABLE/MENU. When a valid table or menu request is received, this module is called to build and format the corresponding display for output.

UPDATE MENU. When valid menu inputs are received, this module is called to update the current menu display. This allows the operator to visually verify the data entries he has made.

OUTPUT HMS DISPLAY. This module outputs the current HMS display to the HMD device.

CHECK MENU INPUT DATA. This module checks the menu inputs to determine if they fall within their respective legal range allowed by the current display. If an error is found a message, informing the operator of the error, is output to the HMD and to the operator console.

PUT HMS MODE ERROR MESSAGE. This module is called to format and output a HMS mode error message to the HMD.

PUT HMS INPUT ERROR MESSAGE. This module is called to format and output a menu input error message to the HMD.





Table XIV  
Parameters for MAKEUP HMD Branch

INPUT		OUTPUT
1	HMD display mode, Vehicle flight mode, Plant dynamics (vehicle, weapons, and target state variables; attack performance), Environment configuration, Cockpit configuration	----
2	----	----
3	----	HMS attitude and position inputs
4	HMS attitude and position inputs, Vehicle state variables	----
5	HMS attitude and position inputs, Vehicle state variables	----
6	HMS attitude and position inputs, Vehicle state variables, Environment configuration	----
7	Vehicle flight mode, HMS attitude and position inputs, Vehicle state variables, Weapons state variables, Attack performance	----
8	HMS attitude and position inputs, Vehicle state variables	----
9	HMS attitude and position inputs, Vehicle state variables, Target state variables, Weapons state variables	----

Table XIV (Cont.)

Parameters for MAKEUP HMD Branch

INPUT	OUTPUT
10 Cockpit configuration	---
11 ---	HMS attitude and position inputs
12 HMS attitude and position inputs	<u>Error flag</u>
13 HMS attitude and position inputs, Recording buffer location	---
14,15,16 HMS attitude and position inputs, Vehicle state variables	---
17 HMS attitude and position inputs, Weapons state variables, Attack performance	---
18 HMS input error message	---

### Module Descriptions for MAKEUP HMD Branch

MAKEUP HMD. Since the HMD is oriented with respect to the operator line-of-sight, this module obtains HMS attitude and position inputs. This module then determines the HMD display mode, activates the corresponding display or displays, and passes the HMS inputs to the appropriate subordinate modules. It is possible and likely that several of the HMD displays will be active at the same time, with the following exceptions: HUD and FSD, VVHD and TERRAIN PORTRAYAL, VVHD and BACKGROUND/ENVIRONMENT. After the display has been built or updated, this module calls a subordinate module to output the display to the HMD device. The HMD graphics data is initialized when this module is first activated.

INITIALIZE HMD SYMBOLOGY. This module is called to initialize the HMD graphics data. The purpose of this initialization is to setup the HMD displays in a form that can be readily updated and output to the display device.

PROCESS HMS INPUTS. This module is called to get HMS attitude and position inputs. These inputs are error checked, recorded, and passed to the superordinate module.

MAKEUP VVHD, MAKEUP TERRAIN PORTRAYAL, MAKEUP BACKGROUND/ENVIRONMENT, MAKEUP HUD, MAKEUP FSD. When requested by the operator, the corresponding module or modules are called to makeup and update the requested displays.



MAKEUP IMAGERY DISPLAYS. This module is called when the target is active and/or when the vehicle weapons are active. It makes or updates the target imagery and/or the weapons envelope imagery.

OUTPUT HMD DISPLAY. This module outputs the current HMD displays to the HMD device.

GET HMS INPUTS. This module is called to get HMS attitude and position inputs from the proper input buffer. The inputs are passed to the superordinate module for processing.

CHECK HMS INPUTS. The HMS attitude and position inputs are error checked by this module. If an error is found, the inputs are discarded and a message, informing the operator of the error, is output to the operator console.

ASSEMBLE TAKEOFF DISPLAY, ASSEMBLE ENROUTE DISPLAY, ASSEMBLE LANDING DISPLAY, ASSEMBLE ATTACK DISPLAY. These modules are called to makeup and update the HUD display that corresponds with the current vehicle flight mode.

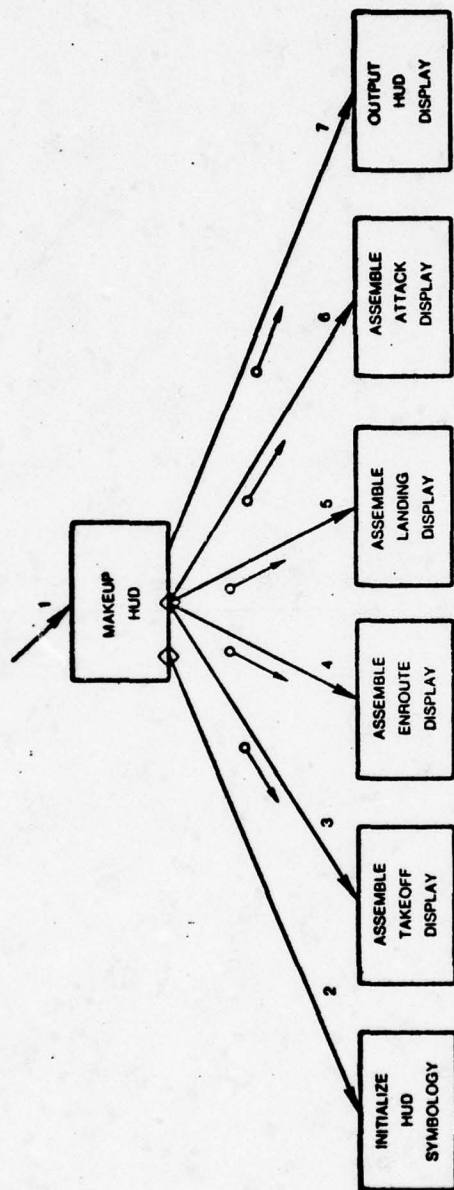


Figure 55. MAKEUP HUD Branch

Table XV  
Parameters for MAKEUP HUD Branch

INPUT	OUTPUT
1 Vehicle flight mode, Vehicle state variables, Weapons state variables, Attack performance, Cockpit configuration	-----
2 -----	-----
3,4,5 Vehicle state variables	-----
6 Weapons state variables, Attack performance	-----
7 Cockpit configuration	-----



### Module Descriptions for MAKEUP HUD Branch

MAKEUP HUD. When called, this module activates the HUD display that corresponds to the vehicle flight mode. After the display has been built or updated, this module calls a subordinate module to output the display to the HUD device. The HUD graphics data is initialized when this module is first activated.

INITIALIZE HUD SYMBOLOGY. This module is called to initialize the HUD graphics data. The purpose of this initialization is to setup the HUD displays in a form that can be readily updated and output to the display device.

ASSEMBLE TAKEOFF DISPLAY, ASSEMBLE ENROUTE DISPLAY, ASSEMBLE LANDING DISPLAY, ASSEMBLE ATTACK DISPLAY.

These modules are called to makeup and update the HUD display that corresponds with the current vehicle flight mode.

OUTPUT HUD DISPLAY. This module outputs the current HUD displays to the HUD device.

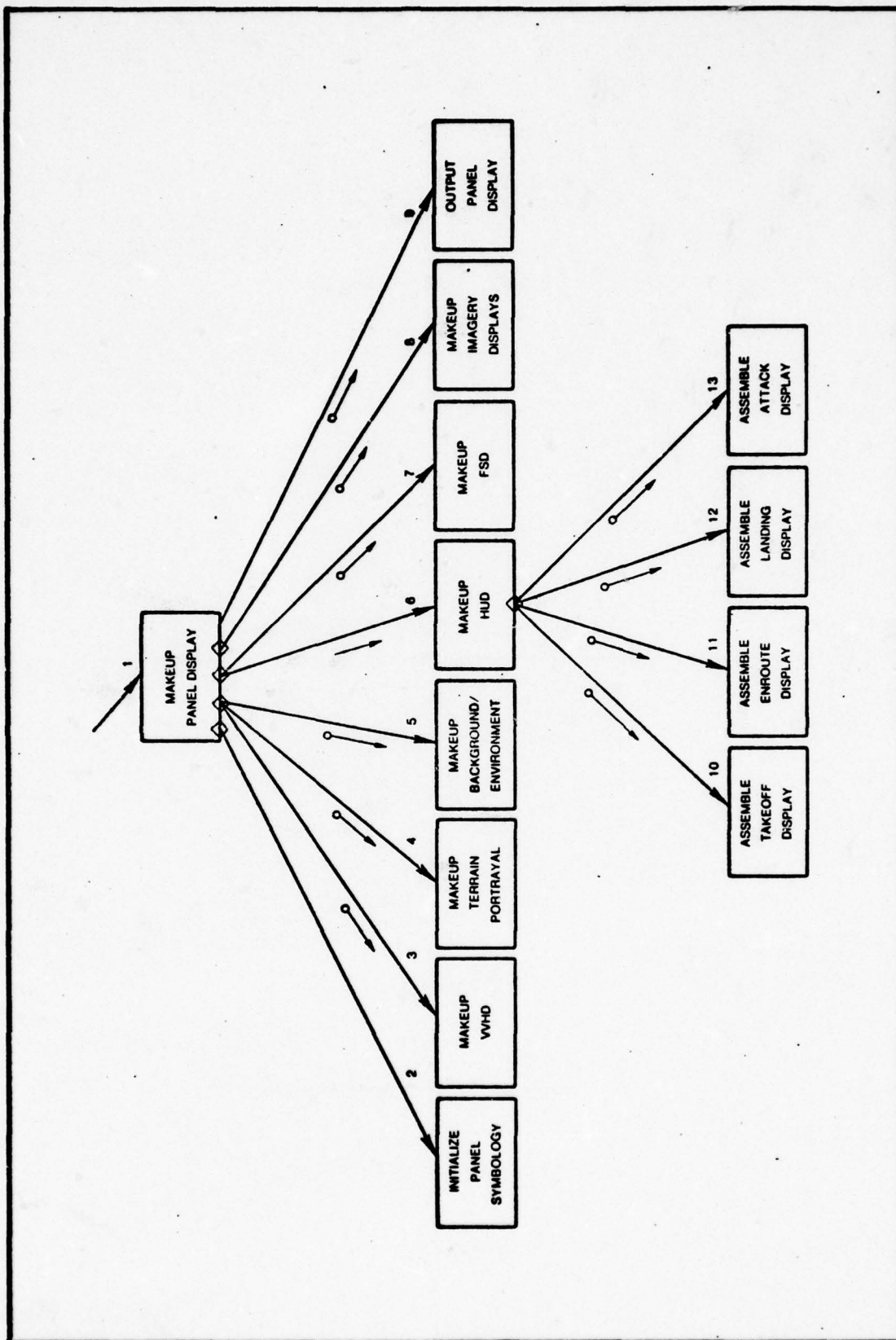


Figure 56. MAKEUP PANEL DISPLAY Branch

Table XVI

## Parameters for MAKEUP PANEL DISPLAY Branch

INPUT		OUTPUT
1	Panel display mode, <u>Vehicle flight mode</u> , Plant dynamics (vehicle, weapons, and target state variables; attack performance), Environment configuration, Cockpit configuration	----
2	----	---
3	Vehicle state variables	---
4	Vehicle state variables	---
5	Vehicle state variables, Environment configuration	---
6	<u>Vehicle flight mode</u> , Vehicle state variables, Weapons state variables, Attack performance	---
7	Vehicle state variables	---
8	Vehicle state variables, Target state variables, Weapons state variables	---
9	Cockpit configuration	---
10,11,12	Vehicle state variables	---
13	Weapons state variables, Attack performance	---



#### Module Descriptions for MAKEUP PANEL DISPLAY Branch

MAKEUP PANEL DISPLAY. This module determines the panel display mode and activates the corresponding display or displays. It is possible and likely that several of the panel displays will be active at the same time, with the following exceptions: HUD and FSD, VVHD and TERRAIN PORTRAYAL, VVHD and BACKGROUND/ENVIRONMENT. After the display has been built or updated, this module calls a subordinate module to output the display to the panel. The panel graphics data is initialized when this module is first activated.

INITIALIZE PANEL SYMBOLOGY. This module is called to initialize the panel graphics data. The purpose of this initialization is to setup the panel displays in a form that can be readily updated and output to the display device.

MAKEUP VVHD, MAKEUP TERRAIN PORTRAYAL, MAKEUP BACKGROUND/ENVIRONMENT, MAKEUP HUD, MAKEUP FSD. When requested by the operator, the corresponding module or modules are called to makeup and update the requested displays.

MAKEUP IMAGERY DISPLAYS. This module is called when the target is active and/or when the vehicle weapons are active. It makes or updates the target imagery and/or the weapons envelope imagery.

OUTPUT PANEL DISPLAY. This module outputs the current panel displays to the panel.

ASSEMBLE TAKEOFF DISPLAY, ASSEMBLE ENROUTE DISPLAY, ASSEMBLE LANDING DISPLAY, ASSEMBLE ATTACK DISPLAY.

These modules are called to makeup and update the panel display that corresponds with the current vehicle flight mode.

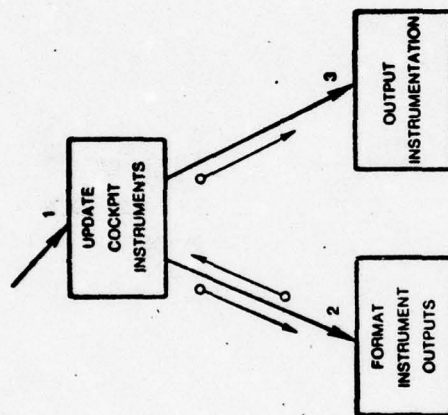


Figure 57. UPDATE COCKPIT INSTRUMENTS Branch



Table XVII  
Parameters for UPDATE COCKPIT INSTRUMENTS Branch

INPUT	OUTPUT
1 Vehicle configuration, Cockpit configuration, Vehicle state variables	----- Formatted instrumentation outputs
2 Vehicle configuration, Vehicle state variables	-----
3 Formatted instrumentation outputs, Cockpit configuration	

Module Descriptions for UPDATE COCKPIT INSTRUMENTS Branch

UPDATE COCKPIT INSTRUMENTS. This module is called to update the cockpit instruments from the respective vehicle state variables. This module calls subordinate modules to format and output the new instrument readings. The instruments to be updated are the clock, airspeed indicator, turn and slip indicator, elapsed time meter, magnetic compass, artificial horizon, directional gyro, vertical speed indicator, altimeter, power indicator, fuel gauge, and gear/flap position indicator.

FORMAT INSTRUMENT OUTPUTS. This module formats the appropriate vehicle state variables for output to the cockpit instruments.

OUTPUT INSTRUMENTATION. This module outputs the formatted instrumentation values to the appropriate hardware instruments.

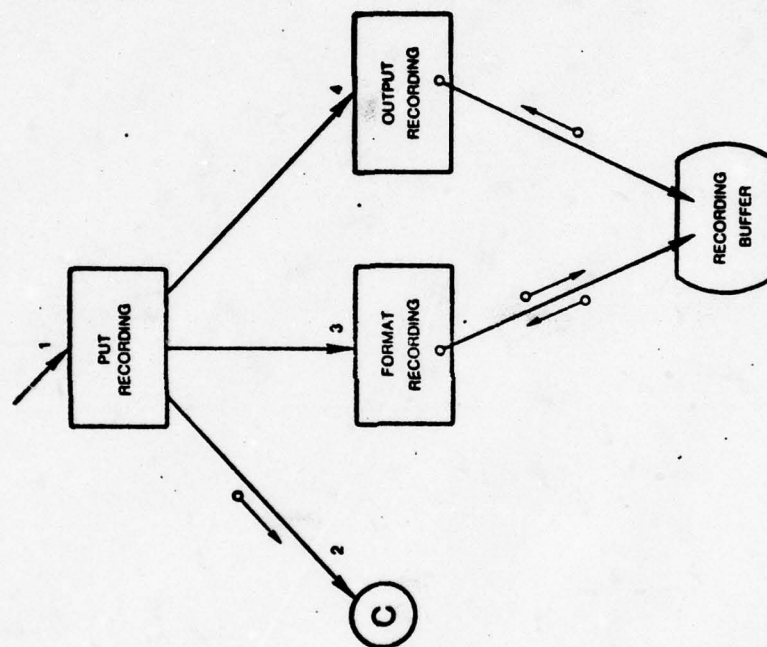


Figure 58. PUT RECORDING Branch



Table XVIII

Parameters for PUT RECORDING Branch

INPUT	OUTPUT
1 Plant dynamics (vehicle, weapons, and target state variables; attack performance)	----
2 Plant dynamics (vehicle, weapons, and target state variables; attack performance)	----
3 ----	----
4 ----	----

#### Module Descriptions for PUT RECORDING Branch

PUT RECORDING. This module is periodically called by the executive to put the current plant dynamics into the recording buffer, and then dump that buffer to a recording device. All operator inputs and outputs, analog and digital inputs, plant state variables, and attack performance are contained in the recording buffer.

FORMAT RECORDING. This module puts the data in the recording buffer into the proper format for output to the recording device.

OUTPUT RECORDING. This module writes the formatted recording data to the recording device.

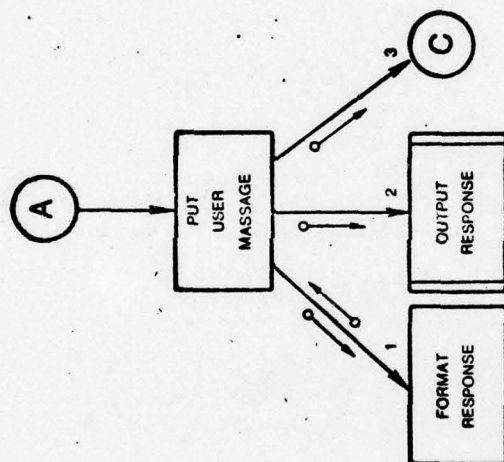


Figure 59. PUT USER MESSAGE Branch



Table XIX  
Parameters for PUT USER MESSAGE Branch

INPUT	OUTPUT
1 Unformatted message	Formatted message
2 Formatted message	-----
3 Formatted message, Recording buffer location	-----

Module Descriptions for PUT USER MESSAGE Branch

PUT USER MESSAGE. This module is called to format and output messages to the operator console. The messages passed to this module are unformatted.

FORMAT RESPONSE. This module formats a message for output to the operator console.

OUTPUT RESPONSE. This is a system supplied module to output data to the operator console.

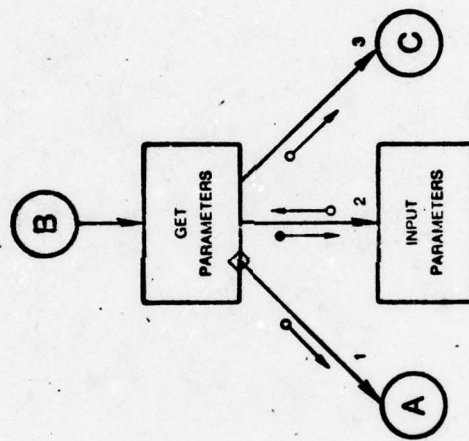


Figure 60. GET PARAMETERS Branch



**Table XI**  
**Parameters for GET PARAMETERS Branch**

INPUT	OUTPUT
1 Parameter request message	<div data-bbox="532 913 552 991">---</div> <div data-bbox="586 627 618 991">Configuration parameters</div> <div data-bbox="651 913 670 991">---</div>
2 <u>Storage device</u>	
3 Configuration parameters, Recording buffer location	

#### Module Descriptions for GET PARAMETERS Branch

GET PARAMETERS. During the set-up of the system configuration, this module is called to either obtain parameters from the operator console or from a specified storage device. When the parameters are to be obtained from the operator console, this module outputs a message to the console requesting the entry of the respective configuration parameters. In both cases the parameters are recorded and passed to the calling superordinate module.

INPUT PARAMETERS. This module obtains input parameters from the operator console or a specified storage device.

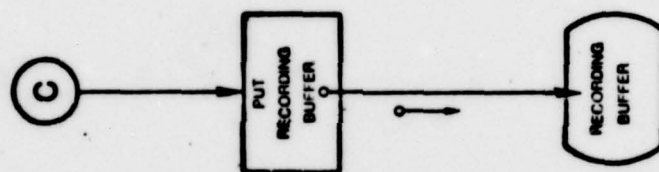


Figure 61. PUT RECORDING BUFFER Branch



Module Descriptions for PUT RECORDING BUFFER Branch

PUT RECORDING BUFFER. The data passed to this module is placed in the recording buffer according to the specified buffer location.

### Summary

This chapter has presented the final bubble chart and structure chart design, with their associated documentation, for the VCASS system software. The preliminary bubble chart was obtained from the formal specifications in Chapter III. From the preliminary bubble chart and the formal specifications a first cut was made of the structure charts. This structure chart was analyzed relative to the evaluation criteria (coupling, cohesion, scope-of-effect, and scope-of-control). Several iterations of the structure-to-bubble chart and the bubble-to-structure chart transforms were necessary to produce the final design.

## V. Design Development

### Introduction

The design of the VCASS system consisted of two main phases: production of the requirements definition and design of the software structure. This chapter is devoted to the discussion of the techniques used in these two phases, some of the problems encountered, and how these problems were solved. The conclusion includes some general observations and recommendations.

### Requirements Definition Phase

The requirements definition is one of the most difficult and most important phases of a development project (Ref 2:5). An analysis procedure used to obtain a complete and comprehensive requirements definition is SofTech's Structured Analysis Design Technique (SADT) (see Ref 5). This technique helps to analyze and document "what" the system is to accomplish. Two sets of diagrams result: one describes the system in terms of activities and the other describes the system in terms of data. When the two sets of diagrams have been developed, they are then cross checked and sequenced to provide the complete and comprehensive requirements definition. It was found that for systems such as this one that are not data oriented, the data model does not lend any significant information to the analysis. It is felt that the amount of time spent in developing the data model was not worth the little added insight it gave. A discussion of the structured analysis technique steps used in authoring the activity diagrams and how they were applied to this project are contained in this section. The same steps were basically applied to authoring the data diagrams. (All



diagrams referenced in the following steps are contained in Chapter III).

Diagram Authoring Steps (Ref 4):

1. Select Child to Decompose. When possible, the first child selected for decomposition is the one which will provide the most in-depth information about the parent. The information obtained by such a decomposition helps in decomposing the other children. For example, in the decomposition of node A1 ('Process User Commands') the child A13 ('Configure System') was chosen to be decomposed first because it would detail what type of inputs had to be processed. From this information it was easier to decompose A11 ('Get Valid Command').

When decomposing a parent, the "depth" should be consistent if possible. This in-depth consistency is not always necessary as can be seen from the structure analysis diagrams in this thesis.

2. Gather Information. Once the child to be decomposed is selected, all available information which has any bearing on this child is collected. In this project the available sources of information consisted of AMRL personnel who are associated with the project and a preliminary document describing a F-16 simulator written by the Air Force Avionics Laboratory (Ref 11).

3. Create a First Draft of the New Diagram. Using the information collected, a list is made of the data to be used by the child. The data list is then grouped according to the relationships between the data. Also created is an activity list based upon all data relationships. An example of the lists created in the drafting of node A13 ('Configure System') is shown below.

### Data List

Vehicle type	}	Vehicle parameters
Vehicle range		
Vehicle load capacity		
Vehicle maximum velocity		
Vehicle stall velocity		
Vehicle weapons		
Weapon types	}	Weapons parameters
Weapon effective range		
Weapon drag coefficient		
Weapon quantities		
Dual or single cockpit	}	Cockpit parameters
Types of cockpit instruments		
Canned target	}	Target parameters
Semi-smart target		
Smart target		
Windspeed	}	Environment parameters
Visibility range		

### Activity List

Configure vehicle  
Configure weapons  
Configure cockpit  
Configure target  
Configure environment

From the activity list, the activity boxes and the connecting data arrows are drawn, giving the first draft of the new diagram. In the development of the activity model, an inability to define input, output, or control indicates a lack of understanding of that particular activity. Therefore, further study of the system is required until a level of understanding is obtained which allows development to continue.

4. Test Diagram Quality. Once the draft of the new diagram is finished, it is tested for viewpoint, purpose, and completeness to ensure that the diagram and its parent are consistent. Other areas of testing are amplification factor and balance.

The amplification factor test of a diagram should ensure the introduction of sufficient information to make the child diagram more informative than the parent, without having the child be complex and difficult to understand. If there is insufficient information in a diagram to make it more informative than its parent, either the diagram should be studied further and redrawn or else the parent should not be decomposed. Node A23 contains such an example; the decomposition of box 2 would have resulted in a node diagram having the same description as the parent (perform a table lookup to maneuver the canned target).

The balance test ensures that all activities and arrows in the diagram are of the same level of detail. When the diagram balance is not maintained at the same level of detail, the diagrams become hard for the reader to follow and to understand. Thus, the major purpose of the structured analysis diagrams to convey and clarify information, may be lost.

5. Write the Text. The text to be appended to a diagram should give the reader a better understanding of what is happening in that diagram. This text should consist of sufficient information to clarify the activity at the described level of detail.

There are two points that aid in writing a good text. First, the text should be written to clarify relationships between inputs, controls, and outputs of the different boxes of that diagram. Second, all names in the text should be consistent with those used on the diagram. Node A2 text is an example of a text that meets the above requirements. When reading the portion of A2 text given below refer to node diagram A2.



'Compute Weapons Plant Dynamics' (4) determines the weapons mode from the weapon inputs (4C3) and the weapon configuration (4C4). The vehicle state variables (4C2) and the target state variables (4C1) are then used to control the particular weapon mode by modifying and updating the weapon state variables (401).

6. Modify Diagram or Parent. Modification of a diagram or its parent is required whenever an error or an inconsistency between the two is found. These errors or inconsistencies are found when testing the diagram for quality or when writing the text. Modification of a diagram or its parent is also required when changes are made to the requirements definition. To correct the inconsistencies, either the diagram, its parent, or both are modified.

#### Software Design Phase

The purpose of the design phase is to define the functions and operations necessary to satisfy the system requirements. Constantine's structured design (see Ref 10) is a technique to describe the system for the programming phase of development. Two of Constantine's basic structured design techniques are used in this project: the transform analysis and the transaction analysis techniques.

The transform analysis technique is used on transform-centered-systems. A transform-centered-system is one in which the major system functions are viewed as central transforms which use the major system inputs to create the major outputs. The purpose of this technique is to identify the primary processing functions of the system, the high-level inputs to those functions, and the high-level outputs (Ref 10:254). A detail discussion of this technique is found in Ref 10.

The transaction analysis technique uses the characteristics of a transaction center to develop a modular software structure. A

AD-A055 226

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 1/4  
SOFTWARE DESIGN FOR A VISUALLY-COUPLED AIRBORNE SYSTEMS SIMULAT--ETC(U)  
MAR 78 W H REEVE, J L STINSON

UNCLASSIFIED

AFIT/GCS/EE/78-6

NL

3 OF 3  
AD  
A055 226



END  
DATE  
FILMED  
7-78  
DDC

transaction is any element of data, control, signal, event, or change-of-state which causes, triggers, or initiates some action or sequence of actions. A transaction center must be able to obtain the transaction, determine its type, dispatch on that type, and complete the processing of each transaction (Ref 10:303). A more detailed discussion of this technique is found in Ref 10. A discussion of the steps used in these two techniques, including examples are contained in this section. These steps are slightly modified from those listed by Constantine. (Ref 10:254-328).

Another structure design technique used in this thesis is that of Hughes Aircraft (Ref 1). This method consists of iterating back and forth between the bubble chart and the structure design chart until both of these are comfortable, reasonable, and functional. With this iteration method, minor changes in the requirement definition can be made at anytime during the iteration process without significant setbacks to the development schedule.

#### Transform Analysis Steps (Ref 10:254-271):

1. Restate the Problem as a Bubble Chart. The first problem encountered is how to make a transition from the structured analysis models to the structure charts. This transition is accomplished through the use of a bubble chart. From the formal requirements definition specified in the structured analysis models, a preliminary bubble chart is drawn to show the flow of data through the system and the transformations on that data. The preliminary bubble chart drawn for this project is shown in Figure 62.

2. Identify the "Afferent" and "Efferent" Data Elements on the Bubble Chart. The initial identification is shown in Figure 62.



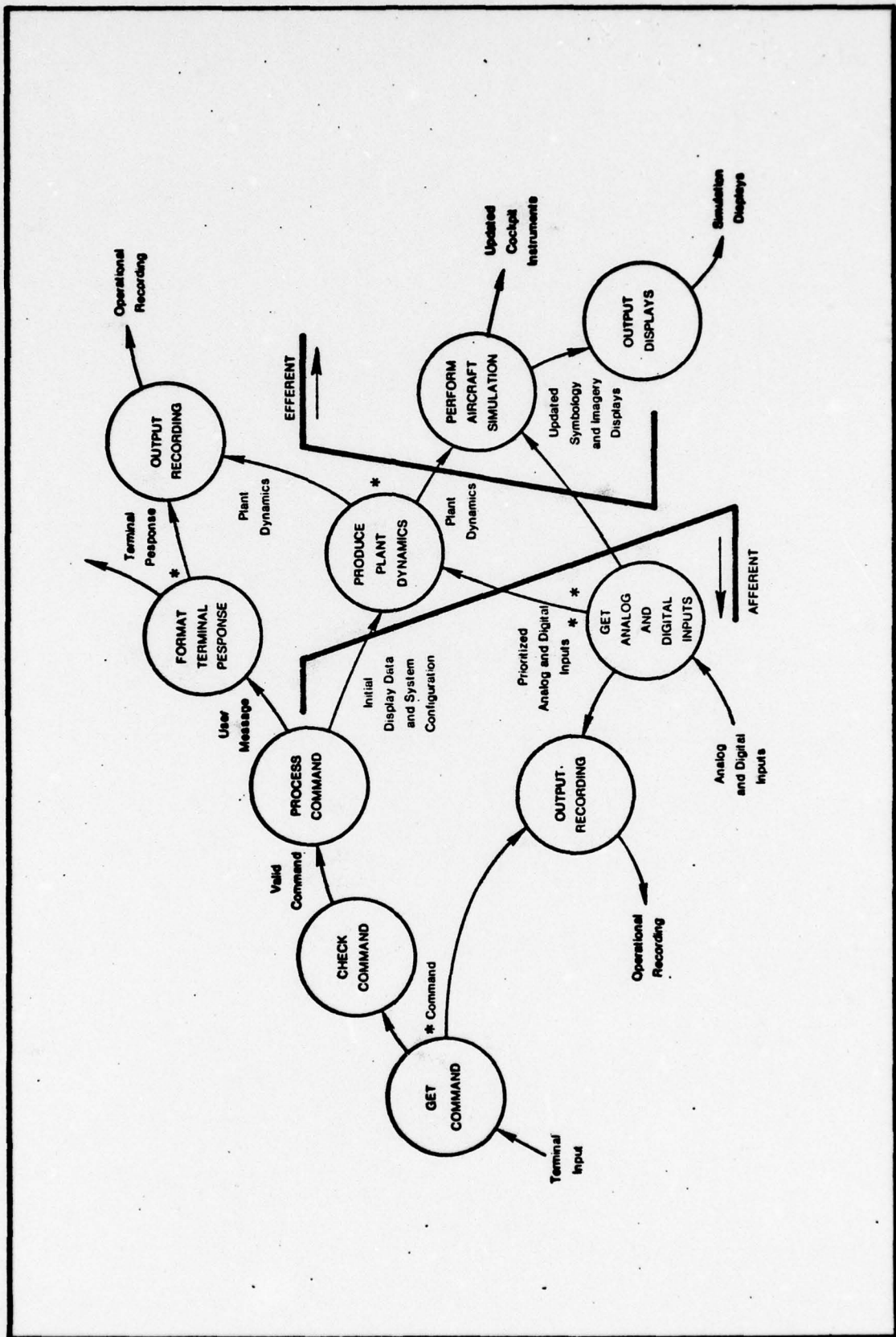


Figure 62. Preliminary Bubble Chart

"Afferent data elements are those high-level elements of data which are furthest removed from physical inputs, yet which still constitute inputs to the system" (Ref 10:262). "Efferent data elements are those furthest removed from the physical outputs which may still be regarded as outgoing" (Ref 10:263). The transforms in the middle between the "afferent" and the "efferent" data elements are designated as "central transforms".

3. Develop a Fully Factored "First-Cut" Structure. First, a main module ('VCASS SUPERVISOR') is defined to call upon subordinates to perform the system tasks. Second, the "afferent" modules ('PROCESS COMMAND' and 'GET ANALOG AND DIGITAL INPUTS') are made subordinate to 'VCASS SUPERVISOR'. Their functions are to deliver the system inputs to their superordinates. Third, the "central transform" module ('PRODUCE PLANT DYNAMICS') is made subordinate to 'VCASS SUPERVISOR'. Its function is to accept input data from its superordinate, transform the input data into the appropriate output data (plant dynamics), and return the output data to the superordinate. Lastly, the "efferent" module ('PERFORM AIRCRAFT SIMULATION') is also made subordinate to 'VCASS SUPERVISOR'. The function of 'PERFORM AIRCRAFT SIMULATION' is to accept the plant dynamics from the supervisor and transform it into display data for output. The results of this "first-cut" top-level factoring is shown in Figure 63. The method used for factoring of these three type of modules was accomplished by the use of the appropriate structured analysis diagrams (Chapter III). Because of the size of the system, it would be impractical to show the complete "first-cut" structure chart. (Note: many of the nodes of the structured analysis diagrams correspond directly as modules in the structured design.)

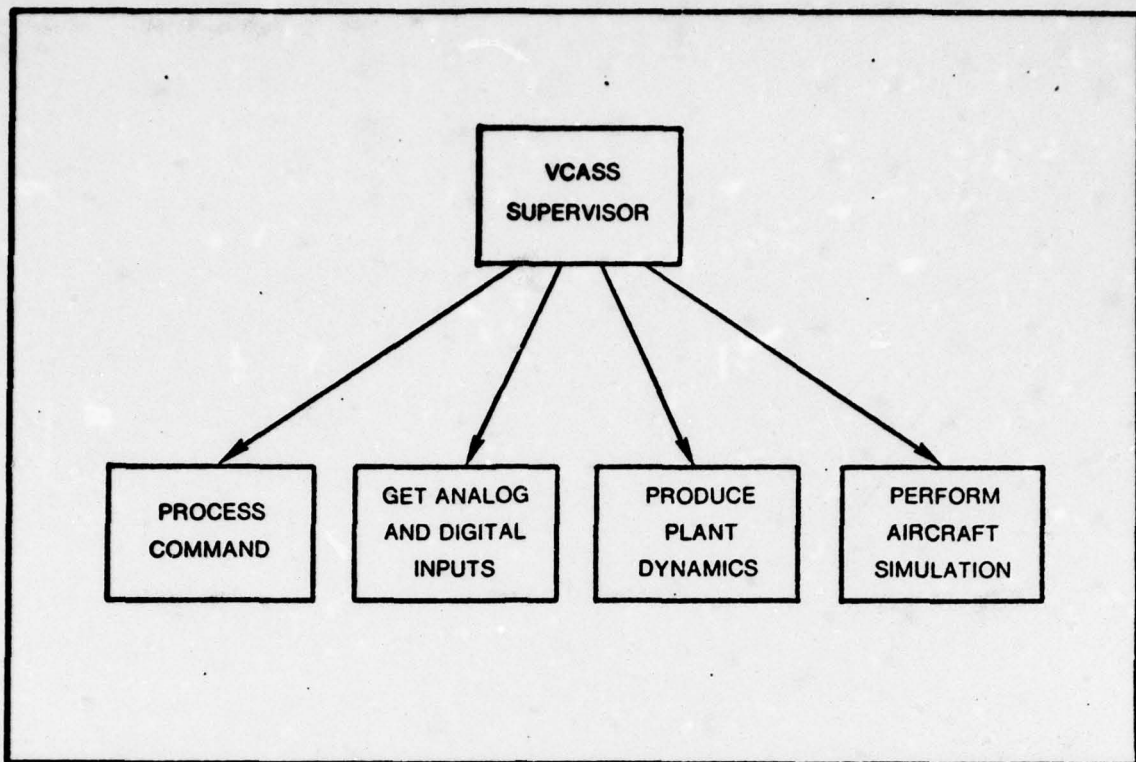


Figure 63. "First-Cut" Top-Level Structure Chart

4. Analyze and Redraw (as necessary) the "First-Cut" Structure Chart. This is accomplished by evaluating the structure chart as to the following attributes: coupling, cohesion, span-of-control, and scope-of-effect.

When the "first-cut" structure was analyzed it became evident that placing the module 'GET ANALOG AND DIGITAL INPUTS' at the top level caused input control data to be passed down several module levels before the data was used. This created unnecessary control coupling throughout the structure. The problem was solved by making each module responsible for obtaining the inputs which it needed. Figure 47 (Chapter IV) illustrates such an example. The vehicle inputs obtained by the modules in this example are obtained indirectly by a call to the module 'GET VEHICLE PARAMETERS', rather than being passed down



from the top of the structure.

Cohesion and coupling are interrelated. The lower the coupling between any pair of modules, the greater the cohesion of the individual modules (Ref 10:144). This was found to be true. When coupling problems were solved, a satisfactory level of cohesion existed within the modules. For example, when the coupling problem, discussed above, was resolved the cohesion of the input modules improved. In the "first-cut" structure, in which the inputs were obtained at the top level through the module 'GET ANALOG AND DIGITAL INPUTS', many of the lower level input modules were logically and communicationally cohesive. (Logical cohesive modules are those that perform a general function and communicational cohesive modules are those which comprise several logical functions operating on some data (Ref 10:153, 161).) This was due to the fact that the input modules at this level had to collect different inputs, group them, and pass them on. After the coupling problem was resolved, by placing the input modules subordinate to the modules which directly uses the input data, the input modules became functionally cohesive. (Functional cohesive modules are those that perform a single specific function and are the most desirable type of modules (Ref 10:170).)

Making the scope-of-effect and scope-of-control coincide was not a significant problem in the design of the VCASS simulator. One place that they do not coincide was in the 'CONFIGURE SIMULATION' branch (Figure 44, Chapter IV); they could have been made to coincide by designing the structure as shown in Figure 64.

The problem with the structure in Figure 64 is that the design is not balanced. Each of the 'CONFIGURE' modules is logically equivalent

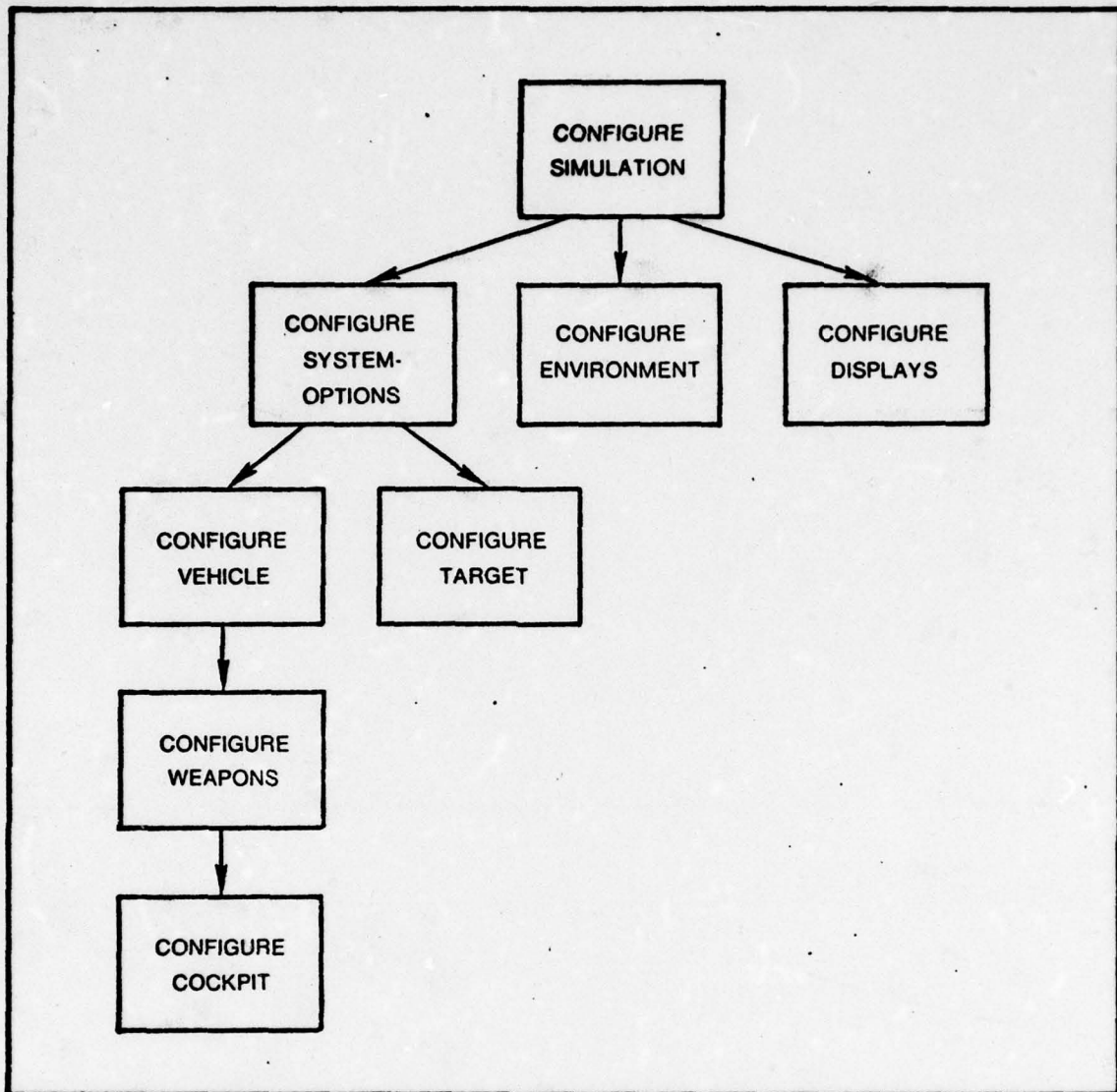


Figure 64. Alternate CONFIGURE SIMULATION Branch

and should not be subordinate to the others. The scope-of-effect which exists between some of the modules is small and therefore is not as important as maintaining a balance in the design. An example of this small scope-of-effect is easily seen between the modules 'CONFIGURE VEHICLE' and 'CONFIGURE WEAPONS'. The vehicle type dictates what type of weapons and how many it can carry, but reveals nothing about the many individual characteristics of the weapons.

5. Iterate Procedure. Using the Hughes Aircraft Method, iterate between the structure chart and the bubble chart until both types are comfortable, reasonable, and functional. This iteration procedure was found to be very useful to "fine tune" the requirements definition throughout the design phase.

A significant problem that was encountered during the iteration process was where to accomplish the operational recording within the structure. This problem was later solved when AMRL specified that all the recording information was to be contained at a single specified buffer location and periodically dumped to a recording device. It was then decided to make the recording, module ('PUT RECORDING') independent from the other modules. Other modules would dump new data into the recording buffer and the recording module would output the buffer when activated by the "supervisor". Thus, the recording module became immediately subordinate to the module 'VCASS SUPERVISOR'.

Transaction Analysis Steps (Ref 10: 307-308):

1. Identify a Transaction Center. When the transform analysis procedure was applied, some of the lower level modules were found to be transaction centered. One such module is 'MAKEUP HMD' (Reference Figure 54, Chapter IV). This module and corresponding figure are used for the example in the following steps and should be referenced.

2. Identify the Transactions and Their Defining Actions.

<u>Transaction</u>	<u>Action</u>
VVHD mode	Makeup the VVHD display
Terrain Portrayal mode	Makeup the Terrain Portrayal display
Background/Environment mode	Makeup the Background/Environment display
HUD mode	Makeup the HUD display
FSD mode	Makeup the FSD display



3. Note Potential Situations Where Modules can be Combined.

In this example each transaction is a separate and distinct mode or state; therefore, none of the modules can be combined. If, in the future some of the displays were to be combined to reduce cockpit clutter for the pilot or if two of the displays were found to be almost similar, then this would be an example where two modules could be combined.

4. Specify a "Transaction" Module for Each Transaction.

From the transaction list in Step 2 the following "transaction" modules were specified to supervise the corresponding transaction.

<u>Transaction</u>	<u>Transaction Module</u>
VVHD mode	MAKEUP VVHD
Terrain Portrayal mode	MAKEUP TERRAIN PORTRAYAL
Background/Environment mode	MAKEUP BACKGROUND/ENVIRONMENT
HUD mode	MAKEUP HUD
FSD mode	MAKEUP FSD

5. Specify a Subordinate "Action" Module. See explanation under

6.

6. Specify Appropriate Subordinate "Detail" Modules. Due to an incomplete requirements definition in the area of displays, the "action" and "detail" modules of this transaction center were not specified. Later, as AMRL further defines the detailed display requirements the "transaction" modules can be expanded to complete steps 5 and 6. The "transaction" module 'MAKEUP HUD' is, also, a transaction center and these same six steps are applied to its design.

Finalizing the Design. After the "final" structure chart is drawn and before programming begins, the associated design documentation should be prepared. This is accomplished by creating an input/output para-

meter list along with a module description of each module. This design documentation will give the programmer a basic understanding of what each module does and how it interfaces with the rest of the system. The programmers should then be able to flow chart and program each module with minimal difficulties.

When the associated design documentation is finished, it, along with the "final" structure charts, are given to the appropriate people for a final design review. During this review, the design is scrutinized and checked to ensure conformity to the established requirements definition. Possible problems are also noted, recommendations made, and remedies proposed. For this project, AMRL conducted the final design review. Their remarks and recommendations were analyzed and the design modified accordingly. The results of this final design review produced the "final" structure chart design contained in Chapter IV.

#### Observations and Recommendations

When answering the questions that arise in defining the formal functional specifications for a proposed system, an organization is forced to make concrete decisions about what they want the system to accomplish. This aids in organizing the varied ideas and concepts of "how" the system is to perform. It was found to be very important to periodically hold system development meetings. These will save time, will facilitate the answering of questions that arise, and will foster the exchange of ideas and recommendations.

Many of the bottom level modules for creating and assembling the displays (Figures 53 and 56) need further decomposition to better define their functions. The modules have not been decomposed

in this thesis because AMRL has not yet defined the detailed display requirements. These modules are so structured that they can be examined and decomposed further, as required, without effecting other modules in the system.

The programmers that implement this design should be familiar with the structure design techniques as contained in Ref. 10. This will help them to better code the different modules. The structure charts should be studied by the programmers to clarify any vague portions of the design. If further decomposition is needed for a module, it should be accomplished before the coding is started. To be consistent with the software life-cycle (discussed in Chapter II) each module should be flowcharted for coding. These flow charts will provide a more rigorous specification for the modules than the formal functional specifications given in Chapter IV.



## Bibliography

1. Jensen, E. "Structured Design." 1975 IR&D Structured Design Methodology, 2: Hughes Aircraft Company, April 1976.
2. Logicon, Inc. Management Guide to Avionics Software Acquisition, Volume I - An Overview of Software Development and Management. Dayton, Ohio: June 1976.
3. McGowan, Clement L. and J. R. Kelly. A Review of Some Design Methodologies. Waltham, Massachusetts: SofTech, Inc., September 1976.
4. Peterson, J.B. Lecture notes taken in E.E. 6.93, Software Engineering. School of Engineering, Air Force Institute of Technology. Wright-Patterson AFB, 1977.
5. Ross, D.T., and K. E. Schoman, Jr. Structured Analysis for Requirements Definition. Waltham, Massachusetts: SofTech, Inc., April, 1976.
6. SofTech, Inc. An Introduction to Structured Analysis and Design Technique. Waltham, Massachusetts: November 1976.
7. SofTech, Inc. Structured Analysis Reader Guide. Waltham, Massachusetts: May 1975.
8. Stevens, W.P., et al. "Structured Design." IBM Systems Journal, 2: 115-139 (1974).
9. Warner, Hollace H. Development of a Synbology Exerciser for Display Generation and Analysis on the Visually-Coupled Airborne Systems Simulator. Unpublished thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, March 1978.
10. Yourdon, E., and L. L. Constantine. Structured Design. New York: Yourdon, Inc., February 1976.
11. Zissos, Stephen N. F-16 Independent Assessment Simulator Product Specification. Wright-Patterson AFB, Ohio: Air Force Avionics Laboratory, September 1976.

## Appendix A

### STRUCTURED ANALYSIS DIAGRAMS

#### Introduction

This appendix contains a brief explanation of the functional analysis phase of structured analysis to aid the reader in understanding the development of the design structure. A more detailed discussion can be found in Ref 7.

#### Structured Analysis

Structured analysis is a comprehensive methodology for performing functional analysis and design. In the functional analysis phase, the emphasis is on analyzing and documenting "what" the system is supposed to do. Two sets of diagrams result: one describes the system in terms of activities, the other describes the system in terms of data. These diagrams are created by decomposing the system into smaller and smaller pieces. The two sets of diagrams are cross-checked and sequenced to provide a model of the system functions.

#### Diagram Syntax

Structured analysis diagrams consist of labeled boxes and arrows for expressing the system activity and data models. Figure 65 illustrates the basic syntax of the models. Inside a box is the name of the activity model or data model. For an activity model the name expresses the action taking place. For a data model the name is a noun or noun phrase expressing the data item.

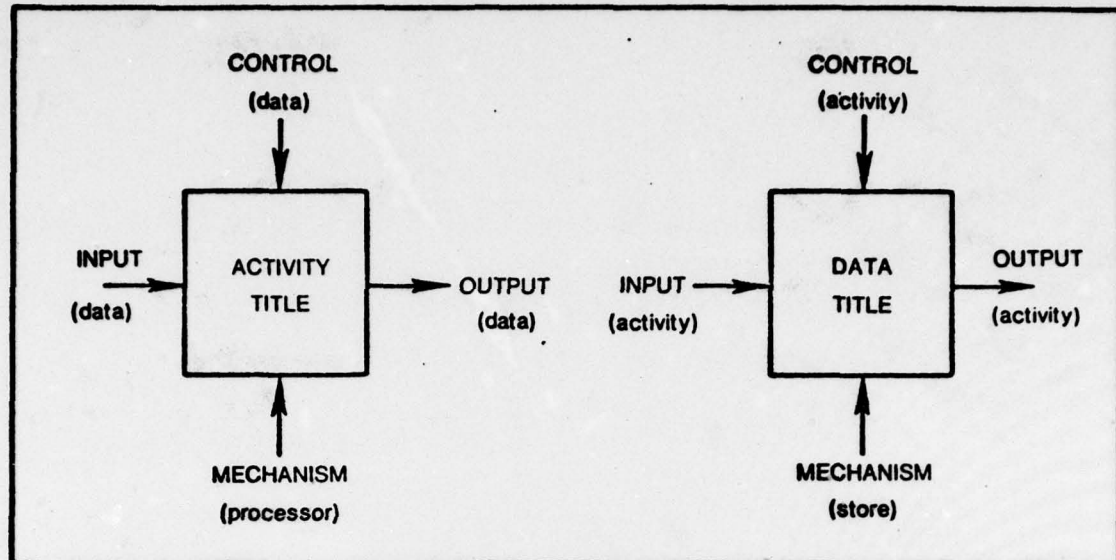


Figure 65. Box/Interface Arrow Conventions

The boxes of the parent diagram are decomposed into more detailed diagrams called children. Each diagram is numbered in a Dewey-decimal manner (Ref 7: 2-3), which represents the parent-child relationship. Diagrams 311, 312, 313, and 314 would be children of diagram 31. Each diagram is referenced as a node.

The boxes of a diagram are connected by arrows which represent the interface between the boxes. The sides of the box defines the kind of arrow(s) which may enter or leave that side of the box.

Four types of arrows represent the kinds of interface. As in reference 7: 3-6, these are:

A. Activity Diagrams:

1. Input: Data transformed by the activity into the output.
2. Output: Data created by the activity.
3. Control: Data used to control the process of converting the input into output.
4. Mechanism: The processor which performs the activity (person computer, program, etc.)



## B. Data Diagrams:

1. Input: Activity which creates the data.
2. Output: Activity which uses the data.
3. Control: Activity which controls the creation or use of the data.
4. Mechanism: The storage device used to hold the data (buffer, computer memory, etc.)

It should be noted that the "mechanism" arrow represents the tool necessary to "realize the box" (Ref 7: 3-4); since it is usually evident from the title of the box the "mechanism" arrow is not always shown.

The "mechanism call" arrow points downward. This indicates a separate system which completely performs the function of the box. In such cases there will be no child diagram of the box and its detail would be found in a separate model of the mechanism. This "mechanism call" is illustrated in Figure 66.

The "multiple branch" (EXCLUSIVE OR) is used to indicate multiple, but not simultaneous, outputs. The "multiple join" indicates multiple, but not simultaneous, inputs. Both conventions are illustrated in Figure 67.

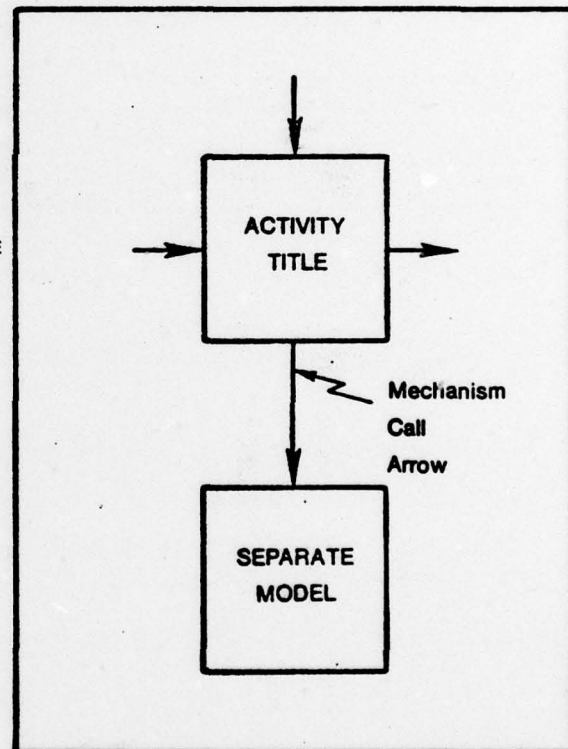


Figure 66. Mechanism Call Arrow

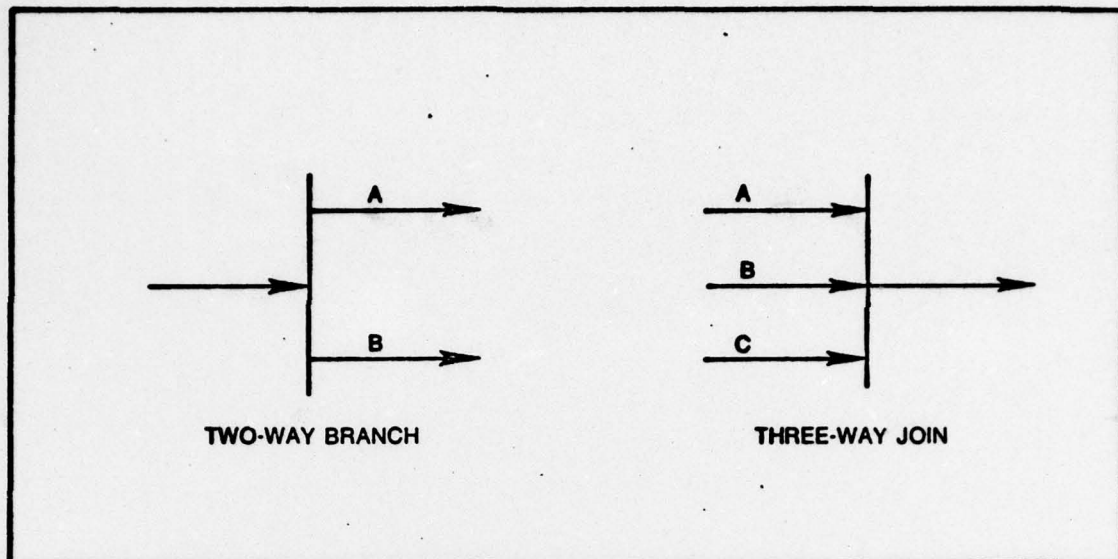


Figure 67. OR Branch and Join Structure

An "ICOM Code" is used to connect arrows across the parent/child boundaries. The name ICOM is derived from the arrow names: inter, control, output, and mechanism. Each boundary crossing arrow (ones which do not have both ends connected to a box) is labeled with its parent-context ICOM code, in addition to its normal label. This aids the reader in locating the matching parent arrow. The "ICOM Code" is written near the unconnected end of the arrow and consists of the letter I, C, O, or M followed by a number. This number gives the relative position that the arrow enters or leaves the side of the parent box. Numbering is done from left to right and top-to-bottom as illustrated in Figure 68. For example, "C2" on an arrow in a child diagram indicates the arrow is the second control arrow entering the parent box.

In the text associated with each diagram, the arrows are identified with an "ICOM code" consisting of a letter (I, O, C or M), a prefix number, and a suffix number. The prefix number refers to the box

within the diagram and the suffix number refers to the top-down or left-right order of the arrow on the box. For example, "2C3" means the third control going into box 2 on the diagram.

Reading Sequence (Ref 7:4-2)

The following sequence is suggested for reading each diagram in a top-down order.

1. Scan only the boxes of the diagram to gain a first impression of its decomposition.
2. Using the parent diagram, rethink the message of the parent, observing the arrows feeding to and from the current diagram.
3. Referring back to the current diagram, see how and where each arrow from the parent context attaches to the factors in the current diagram; using ICOM codes.
4. Consider the internal arrows of the current diagram to see how it works in detail. Consider the boxes from top to bottom and from left to right. Examine the arrows by going clockwise around each box.
5. Finally, read the text of the current diagram to confirm or to alter the interrelation gained from consideration of the diagrams themselves.

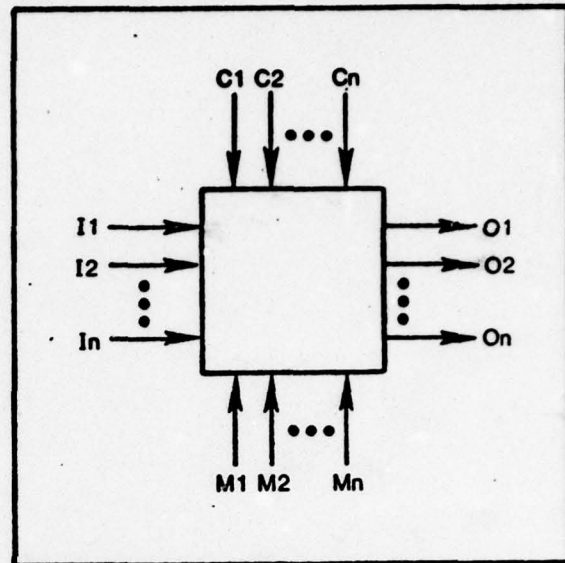


Figure 68. ICOM Numbering Convention



## Appendix B

### STRUCTURED DESIGN CHARTS

#### Introduction

This appendix contains a brief explanation of structured design in order that the reader may understand the development and the reading of the structured charts. A more detailed discussion can be found in Ref 9.

#### Structured Design

Structured design is a set of general program design considerations and techniques for making coding, debugging, and modification easier, faster, and less expensive by reducing complexity (Ref 8: 115). It simplifies the software by dividing it into modules in such a way that the modules can be implemented and modified with minimal effect on other parts of the system.

Structured design starts by stating a problem as a data flow graph, or bubble chart. A first cut is obtained by factoring the bubble chart into a tree structure. The resulting design is then evaluated.

The criteria for evaluating the design are coupling and cohesion. Coupling is a measure of the relationship between modules, and cohesion is a measure of the relationships among elements in the same module. The objective is to minimize coupling and to maximize cohesion. Other criteria to be considered are the scope-of-effect and the scope-of-control. Scope-of-effect is the collection of all modules containing any processing that is conditional upon a particular decision. Scope-of-control is a module and all of its subordinates. The objective is

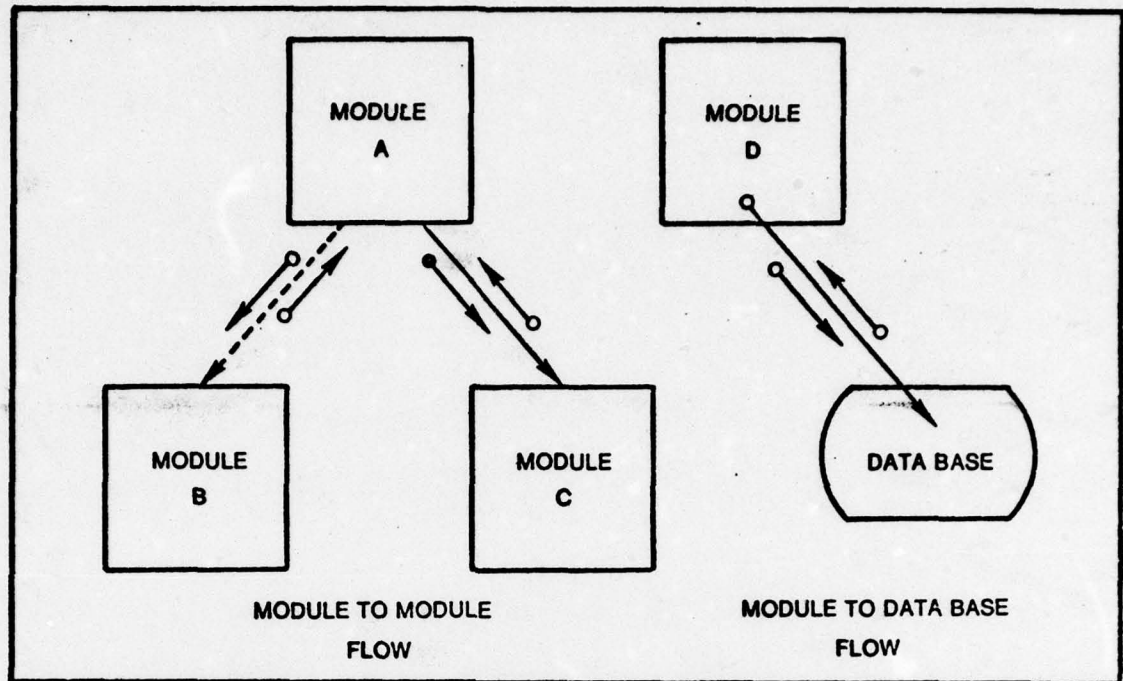


Figure 69. Information Flow

to have the scope-of-effect and the scope-of-control coincide.

#### Structured Chart Syntax

Structured charts consist of modules connected either to modules and/or to data bases by information flow lines or by connection lines. Figure 69 illustrates the basic syntax for connecting modules and data bases. The solid line represents a normal connection. A dashed line represents a transfer of control which takes place automatically, asynchronously or concurrently with established processes. This includes program interrupts and parallel subroutine calls. (Note that the data base is designated by a rectangle with curved sides.) Connections are ordered left-to-right as they emerge from the referencing module in the same order that they would usually be accessed. The arrows adjacent to any connection indicate the direction of the flow of information.

The three types of arrows used are illustrated in Figure 70. An arrow with a small circle on its tail always denotes data. An arrow with a dot (point) on its tail always denotes control. A plain arrow denotes either control, data or both.

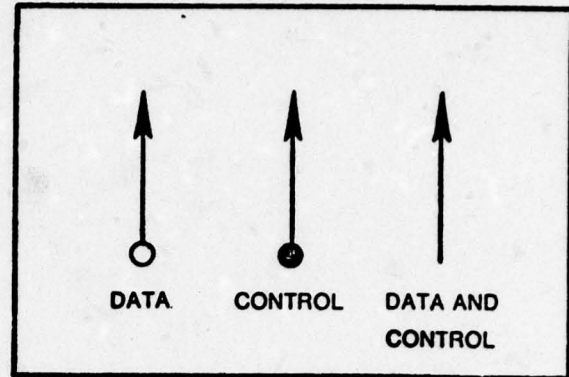


Figure 70. Information Arrow Types

Conditional access to intermodular connections is shown by enclosing the point(s) of reference in a diamond (decision symbol) as illustrated in Figure 71. When intermodular references are used repeatedly within an iterative procedure (loop), the beginnings of the connection(s) are encompassed by a half loop as shown in Figure 71.

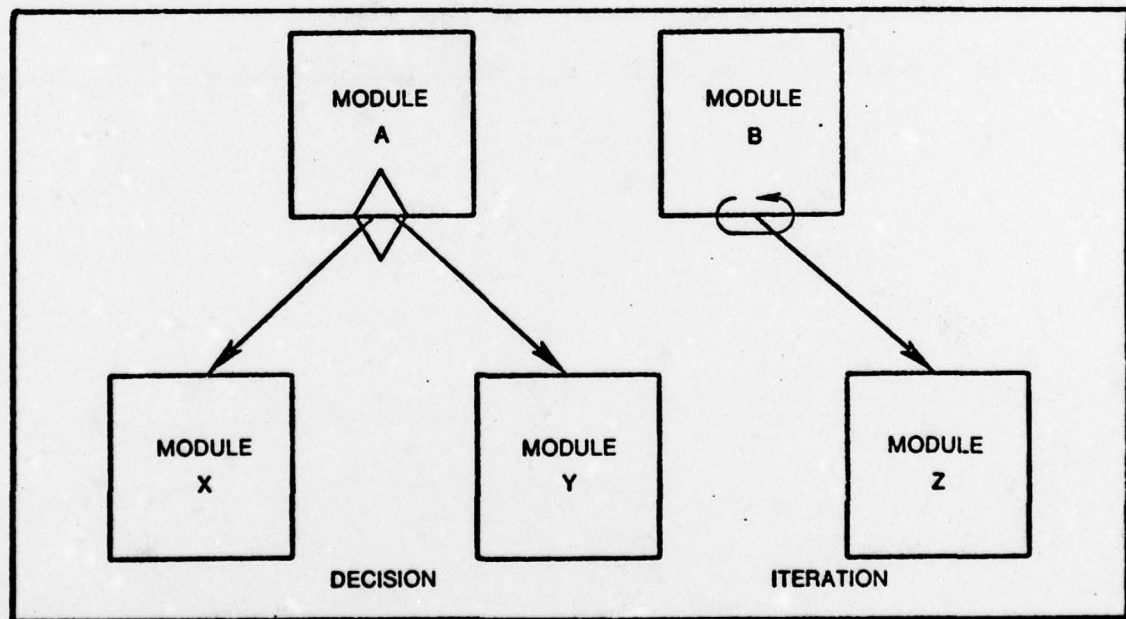


Figure 71. Decision and Iteration



## VITA

PII Redacted

William H. Reeve was [REDACTED].

He graduated from high school in Columbus, Ohio in 1966 and attended Utah State University from which he received a Bachelor of Science Degree in Computer Science. Upon graduation in June 1970, he received a commission in the U. S. Air Force through the ROTC program. For the first three and one half years he served as a computer programmer and systems analyst at the 24th NORAD Region at Malmstrom AFB, Montana. He then served for two and one half years as a computer programmer at HQ NORAD, Ent AFB, Colorado until entering the School of Engineering, Air Force Institute of Technology, in September 1976.

[REDACTED]

## VITA

PII Redacted

Jerry L. Stinson was [REDACTED].

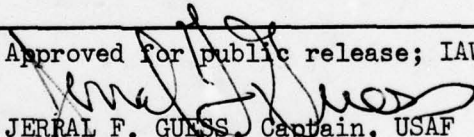
He graduated from Aztec High School in 1965. He went one semester to the New Mexico Institute of Mining and Technology, Socorro, New Mexico. He then transferred to Brigham Young University, Provo, Utah. After receiving two Bachelor of Science Degrees (Mathematics and Computer Science) in June 1972, he was commissioned through the Air Force ROTC Program. He served four years as a Minuteman Missile Combat Crew Member and as an Alternate Positive Control Code Custodian at the 91st Strategic Missile Wing (SAC), Minot AFB, North Dakota. He then entered the School of Engineering, Air Force Institute of Technology in September 1976.

[REDACTED]



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER <b>AFIT/GCS/EE/78-6</b>	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) <b>SOFTWARE DESIGN FOR A VISUALLY-COUPLED AIRBORNE SYSTEMS SIMULATOR (VCASS)</b>		5. TYPE OF REPORT & PERIOD COVERED <b>MS Thesis</b>
7. AUTHOR(s) <b>William H. Reeve                      Jerry L. Stinson Captain                      USAF                      Captain                      USAF</b>		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS <b>Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433</b>		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS <b>6570 Aerospace Medical Research Laboratory (AFSC) Wright-Patterson AFB, Ohio 45433</b>		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE <b>March 78</b>
		13. NUMBER OF PAGES <b>214</b>
		15. SECURITY CLASS. (of this report) <b>Unclassified</b>
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) <b>Approved for public release; distribution unlimited</b>		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES <b>Approved for public release; IAW AFR 190-17</b>  <b>JERRAL F. GUESS, Captain, USAF</b> <b>Director of Information</b>		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <b>Structured Analysis                      Aircraft Simulator Structured Design                      Software Design Bubble Chart Structure Chart</b>		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) <b>This thesis contains an analysis of a Visually Coupled Airborne Systems Simulator (VCASS) and the design of the software for this system. The design is developed in three steps. First, an informal requirements definition is written to establish the viewpoint and the purpose on which the analyst bases his design. This requirements definition explains why the simulator is to be created and what it is to do. Second, a top-down strategy called "structured analysis" is applied to obtain a formal requirements definition.</b>		

DD FORM 1473  
1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

(continued)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)



**UNCLASSIFIED**

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

The structured analysis is presented in a blueprint-type language consisting of activity and data models. These models represent graphically the functions performed by the simulator and the information upon which those functions act. Third, a design is obtained through a structured design methodology consisting of "transform analysis" and "transaction analysis" techniques. The structure charts drawn during the analysis phase reveal system characteristics which illustrate design quality. The activity model is used to make a successful transition from a top-down analysis to a structured design which can be evaluated. The resulting simulator design, with minor revisions, satisfies the design goals established for the project. The methodologies used are highly recommended for the analysis and design of any software system.

**UNCLASSIFIED**

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)